

سطح پیشرفته



# برنامه‌نویسی مدرن C++ همراه با کتابخانه Qt

شامل فناوری QML و Qt Quick

نویسنده : کامبیز اسدزاده

بنام خدا

برنامهنویسی مُدرن C++ همراه با کتابخانه Qt

نویسنده : کامبیز اسدزاده

# پیش گفتار

## دنیای خود را چگونه با **کیوت** بسازیم؟!

با توجه به توسعه روزافزون فناوری، دنیای نرم افزاری همگام با آن با سرعت بسیار زیادی در حال پیشرفت و توسعه است. ما برای رسیدن به این مسیر باید به فکر تولید و توسعه محصول با کیفیت همراه با اقدامات کلیدی باشیم تا این محصول هماهنگ با استانداردهای جهانی باشد. برای این نیاز است تا این استانداردها را بررسی و در درون پروژه‌های خود مورد استفاده قرار دهیم. کیوت به عنوان یک چهارچوب قدرتمند یکی از بهترین و پیشتازترین ابزارهای موجود در دنیای برنامه‌نویسی است که با تمرکز بر روی مباحث تولید محصولی اساساً بر پایه تجربه‌کاربری و رابط‌کاربری پیشرفته همراه با پشتیبانی از قدرتمندترین زبان برنامه‌نویسی، نتیجه‌ای مطلوب را در مسیر توسعه محصول نرم افزاری شما فراهم می‌کند. توجه داشته باشید برنامه‌نویسی صرفاً نوشتمن کد منطقی و برقراری ارتباط با داده‌ها و حل مسائل مربوط به آن نیست! علاوه بر حل مشکل، برقراری ارتباط با احساسات کاربر و ایجاد یک تجربه و تعامل خوب بسیار مهم است.

باید توجه داشت که زمان، هزینه، سرعت و کیفیت همه باهم مهم هستند و برای به حداکثر رساندن درجه کیفیت هر یک از این مولفه‌ها باید از بهترین روش‌های ممکن استفاده کرد که شامل مواردی همچون چند-سکویی، ابری، تجربه‌کاربری، رابط‌های برنامه‌نویسی، کتابخانه‌ها و غیره... می‌باشند و برای رسیدن به آن‌ها کافی است یک زبان مهم و پایه همراه با چند زبان فرعی و فناوری‌های مرتبط با یکدیگر را به عنوان ابزار در اختیار داشته باشیم.

برای اینکه بتوانید آزادانه برنامه‌نویسی کنید زبان سی‌پلاس‌پلاس (C++) گزینه مناسبی است که به تنها یکی می‌تواند بهترین قابلیت‌ها را برای لایه‌های نرم افزاری در بخش‌های **بک‌اند** و سفارشی سازی **فرانت‌ايند** در اختیار شما قرار دهد. همراه با کتابخانه STL (استاندارد)، کتابخانه Qt یک پیشنهاد ویژه است که با پشتیبانی از فناوری **Qt Quick** به شما اجازه برنامه‌نویسی سریع و مدرن را در بخش طراحی رابط کاربری می‌دهد. البته فراموش نکنید این کتابخانه‌ها تنها دو گزینه از کتابخانه‌های بی‌شمار C++ می‌باشند. 😊

امروزه بیشترین محصولات مُدرن تحت فناوری‌های برخطه توسعه پیدا می‌کنند که با سرعت بسیار چشمگیری در حال رشد هستند اما یکی از مشکلاتی که در سیستم آموزشی موجود است این است که بیشتر در زمینه کتاب‌های درسی برای گذراندن واحدهای دانشگاهی مناسب هستند و نباید انتظار داشت برای توسعه استارت‌اپ‌ها و کسب‌وکارهای موفق و جهانی شدن محصول از آن‌ها بهره‌مند شویم، چرا که در بازار واقعی کار هیچ استفاده‌ای نخواهد داشت. زبان سی‌پلاس‌پلاس برخلاف ذات واقعی و کاربردی که دارد بسیار کم رنگ معرفی شده است که هدف این کتاب معرفی و شفاف‌سازی راهکارهای پیشرفته‌ای مانند کیوت است که علاقه‌مندان به این زبان را بیشتر آگاه سازد.

برخلاف کتاب‌های درسی در زمینه برنامه‌نویسی که معمولاً در رابطه با ساختار زبان و قوانین مرتبط با آن اشاره شده است در این کتاب هدف آشنایی و نشان دادن روش صحیح برای طراحی و تولید اجزاء مورد نیاز در ظاهر برنامه اشاره شده است. لذا برای درک بهتر باید در رابطه با زبان برنامه‌نویسی C++ اطلاعاتی را از قبل داشته باشید. در این کتاب در رابطه با نحوه ایجاد یک محصول مدرن و برقراری ارتباط بین **بک‌اند** و **فرانت‌ايند** تحت کتابخانه Qt برای زبان سی‌پلاس‌پلاس توضیح داده شده است که شما را با آن آشنا می‌سازد. فراموش نکنید زبان سی‌پلاس‌پلاس نه تنها بسیار غنی است بلکه دارای کتابخانه‌های وسیعی است که برخی از آن‌ها مانند Qt در زمینه طراحی محصولات مختلف متمرکز شده‌اند. اما توجه کنید تعداد آموزش‌های موجود در این زمینه انگشت شمار است. بنابراین باید شگردهای خاصی برای توسعه بیشتر در این زمینه ابداع کرد.

# آموزش جامع نحوه استفاده از کتابخانه Qt همراه با فناوری Qt Quick در زبان برنامه‌نویسی C++

## شناسنامه کتاب

عنوان کتاب	برنامه‌نویسی C++ همراه با کتابخانه کیوت ۵
سطح	پیشرفتی
تاریخ آخرین ویرایش	۱۳۹۸/۰۷/۲۰
مولف / نویسنده	کامبیز اسدزاده
ناشر	مولف (کامبیز اسدزاده)
نوبت و تاریخ انتشار	اواخر - بهار ۱۳۹۶ - ۲۰۱۷ میلادی
تعداد صفحات محتوا	۴۳۷ صفحه
وب سایت	<a href="http://kambizasadzadeh.com">http://kambizasadzadeh.com</a> <a href="https://iostream.ir">https://iostream.ir</a>
محل نشر	ارومیه: کامبیز اسدزاده، ۱۳۹۶
پست الکترونیکی / انتقادات و پیشنهادات	kambiz.ceo@gmail.com
رده بندی کنگره	QA/۹۸ الف ۱۳۹۶ ۹/۷۶
رده بندی دیویی	۱۳۳/۰۰۵
شماره شابک	۹۷۸-۶۰۰۰-۰۴-۸۰۰۷-۳
شماره کتاب شناسی ملی	۴۷۷۴۴۶۰
قیمت	۶۵.۰۰۰ تومان

## فهرست

### فصل اول

۱	مقدمه بر زبان C++	۱
۱	• برخی از قابلیت‌ها	۱
۱	• ساختار برنامه در C++	۱
۱	• کتابخانه‌ها	۱
۱	• فرق بین C و C++	۱
۲	• ویژگی‌های معرفی شده در C++ ویرایش‌های ۱۱، ۱۴ و ۱۷	۲
۶	• کامپایلرهای C++ و وضعیت آن‌ها	۶
۱۱	• ساختار اسناد C++ در پروژه	۱۱
۱۲	• کاربرد این زبان در کجاست؟	۱۲
۱۵	• استانداردهای زبان	۱۵
۱۶	• مقدمه کیوت (Qt)	۱۶
۱۸	• معرفی کیوت (Qt) ۵.۱۳	۱۸
۲۲	آشنایی با محیط توسعه، نصب و راهاندازی همراه با پیکربندی کیت (Kit) در آن	۲۲
۲۳	• نصب و راهاندازی محیط Qt	۲۳
۲۷	• پیکربندی کیت‌ها در macOS	۲۷
۳۰	• پیکربندی کیت‌ها در Linux	۳۰
۳۲	• پیکربندی کیت‌ها در Windows	۳۲
۳۵	• معرفی محیط توسعه کیوت کریتور (Qt Creator) نسخه ۴	۳۵
۳۷	• پیکربندی و تنظیمات مربوط به ساخت برای پلتفرم‌های مختلف	۳۷
۴۲	• معرفی مجوز‌های Qt و نحوه استفاده از مناسب‌ترین مجوز	۴۲
۴۵	• لوگوهای نشانگر ساخته شده با Qt	۴۵
۴۵	• پشتیبانی از انواع پلتفرم‌ها	۴۵
۴۶	• پشتیبانی از انواع معماری‌ها	۴۶

۴۸	شرایط و قوانین لازم جهت انتشار اپلیکیشن در فروشگاه iTunes یا همان (Apple Store)	•
۴۸	شرایط و قوانین لازم جهت انتشار اپلیکیشن در فروشگاه Windows Store	•
۴۸	شرایط و قوانین لازم جهت انتشار اپلیکیشن در فروشگاه Google Play	•
۴۹	شرایط و قوانین اختصاصی برنامه تحت Qt جهت انتشار و پذیرش در فروشگاه‌های مرتبط	•

## فصل دوم

۵۰	معرفی فناوری Qt Quick ویرایش ۲	
۵۰	معرفی زبان کیوامل (QML) ویرایش ۲	
۵۱	آشنایی با سبک - سینتکس (Syntax) زبان QML	•
۵۴	روش اعلام یا اظهار یک شیء در QML	•
۵۵	اشیاء فرزند (Child-Object) در QML	•
۵۶	سبک و روش اعمال اظهار نظر (Comment) در QML	•
۵۷	صفتهاي اشیاء در QML	•
۶۶	پشتیبانی از جاوااسکریپت (JavaScript) و ترکیب آن با QML	•
۶۷	روش استفاده از جاوااسکریپت در سند QML	•
۷۳	روش‌های ترکیب C++ و استفاده از آن در سند QML	•

## فصل سوم

۸۲	معرفی انواع پروژه‌ها تحت فناوری کیوت کوئیک (Qt Quick)	
۸۳	معرفی پروژه از نوع Qt Widget Application	•
۸۳	معرفی پروژه از نوع Qt Console Application	•
۸۳	معرفی پروژه از نوع Qt for Python - Empty	•
۸۳	معرفی پروژه از نوع Qt for Python - Window	•
۸۴	معرفی پروژه از نوع Qt Quick Application - Empty	•
۸۴	معرفی پروژه از نوع Qt Quick Application - Scroll	•
۸۴	معرفی پروژه از نوع Qt Quick Application - Stack	•
۸۴	معرفی پروژه از نوع Qt Quick Application - Swipe	•
۸۵	آغاز ایجاد پروژه تحت C++ و Qt Quick	
۹۰	ساده‌ترین برنامه	
۹۱	معرفی کلاس QApplication	•

- معرفی کلاس QQmlApplicationEngine
- معرفی کلاس QCoreApplication
- معرفی تابع exec در پروژه

## فصل چهارم

- انواع کنترل‌ها، منو‌ها و دیگر آبجکت‌ها
- معرفی انواع QML پایه در فناوری Qt Quick
    - نوع date
    - نوع color
    - نوع font
    - نوع matrix4x4
    - نوع point
    - نوع quaternion
    - نوع rect
    - نوع size
    - نوع vector2d
    - نوع vector3d
    - نوع vector4d
  - معرفی انواع اشیاء QML در فناوری Qt Quick
    - معرفی Accessible
    - معرفی AnchorAnimation
    - معرفی AnchorChanges
    - معرفی AnchorImage
    - معرفی AnimatedSprite
    - معرفی Animation
    - معرفی AnimationController
    - معرفی Animator
    - معرفی Behavior
    - معرفی BorderImage
    - معرفی Context2D
    - معرفی Canvas
    - معرفی CanvasGradient
    - معرفی CanvasImageData

۱۲۱	معرفی CanvasPixelArray	○
۱۲۱	معرفی CanvasColorAnimation	○
۱۲۱	معرفی Column	○
۱۲۲	معرفی DoubleValidator	○
۱۲۴	معرفی Drag	○
۱۲۴	معرفی DragEvent	○
۱۲۴	معرفی DropArea	○
۱۲۴	معرفی EnterKey	○
۱۲۵	معرفی Flickable	○
۱۲۶	معرفی Flipable	○
۱۲۸	معرفی Flow	○
۱۲۹	معرفی FocusScope	○
۱۳۰	معرفی FontLoader	○
۱۳۲	معرفی FontMetic	○
۱۳۳	معرفی Gradient	○
۱۳۴	معرفی GridMesh	○
۱۳۵	معرفی GridView	○
۱۳۸	معرفی Image	○
۱۴۱	معرفی IntValidator	○
۱۴۳	معرفی Item	○
۱۴۶	معرفی ItemGraResult	○
۱۴۹	معرفی KeyEvent	○
۱۴۹	معرفی KeyNavigation	○
۱۵۱	معرفی Keys	○
۱۵۱	معرفی LayoutMirror	○
۱۵۳	معرفی ListView	○
۱۵۶	معرفی Loader	○
۱۵۷	معرفی MouseArea	○
۱۶۲	معرفی MouseEvent	○
۱۶۴	معرفی MultiPointTouchArea	○
۱۶۵	معرفی NumberAnimation	○
۱۷۵	معرفی OpacityAnimator	○
۱۷۷	معرفی ParallelAnimation	○

۱۷۹	.....	معرفی GraphicInfo	○
۱۸۳	.....	معرفی ParentAnimation	○
۱۸۴	.....	معرفی ParentChange	○
۱۸۸	.....	معرفی Path	○
۱۸۸	.....	معرفی PathAnimation	○
۱۸۸	.....	معرفی PathView	○
۱۹۰	.....	معرفی PauseAnimation	○
۱۹۴	.....	معرفی PropertyAction	○
۱۹۶	.....	معرفی PropertyChanges	○
۱۹۶	.....	معرفی Rectangle	○
۱۹۸	.....	معرفی RegExpValidator	○
۱۹۹	.....	معرفی Repeater	○
۲۰۳	.....	معرفی Rotation	○
۲۰۵	.....	معرفی RotationAnimation	○
۲۰۵	.....	معرفی RotationAnimator	○
۲۰۷	.....	معرفی Row	○
۲۰۷	.....	معرفی Scale	○
۲۰۲	.....	معرفی ScaleAnimator	○
۲۰۳	.....	معرفی SecuentialAnimation	○
۲۰۸	.....	معرفی ShaderEffect	○
۲۰۹	.....	معرفی ShaderEffectSource	○
۲۱۰	.....	معرفی Shortcut	○
۲۱۱	.....	معرفی SmoothedAnimation	○
۲۱۲	.....	معرفی SpringAnimation	○

## فصل پنجم

۲۱۴	.....	● معرفی انواع کنترل‌های 2 Qt Quick Controls 2	
۲۱۶	.....	کنترل AbstractButton	○
۲۱۷	.....	کنترل ApplicationWindow	○
۲۱۷	.....	کنترل BusyIndicator	○
۲۱۸	.....	کنترل Button	○
۲۲۰	.....	کنترل ButtonGroup	○
۲۲۱	.....	کنترل CheckBox	○

۲۲۳.....	CheckDelegate	○
۲۲۴.....	ComboBox	○
۲۲۴.....	Container	○
۲۲۶.....	Control	○
۲۲۷.....	Dial	
۲۲۸.....	Drawer	○
۲۳۰.....	Frame	○
۲۳۲.....	GroupBox	○
۲۳۳.....	ItemDelegate	○
۲۳۴.....	Label	○
۲۳۵.....	Menu	○
۲۳۶.....	MenuItem	○
۲۳۷.....	Page	○
۲۳۷.....	PageIndicator	○
۲۳۹.....	Pane	○
۲۴۰.....	Popup	○
۲۴۱.....	ProgressBar	○
۲۴۲.....	RadioButton	○
۲۴۵.....	RadioDelegate	○
۲۴۶.....	RangeSlider	○
۲۴۷.....	ScrollBar	○
۲۴۹.....	ScrollIndicator	○
۲۵۰.....	Slider	○
۲۵۱.....	SpinBox	○
۲۵۴.....	StackView	○
۲۵۷.....	SwipeDelegate	○
۲۵۹.....	SwipeView	○
۲۶۳.....	Switch	○
۲۶۴.....	SwitchDelegate	○
۲۶۵.....	TabBar	○
۲۶۷.....	TabButton	○
۲۶۸.....	TextArea	○
۲۶۹.....	TextField	○

۲۷۰	○ کنترل ToolBar
۲۷۱	○ کنترل ToolButton
۲۷۳	○ کنترل ToolTip
۲۷۴	○ کنترل Thumbler

## فصل ششم

۲۷۷	● معرفی Qt Quick Dialog (دیالوگ های انتخاب رنگ، فایل، فونت و پیغام)
۲۷۹	● معرفی Color Dialog
۲۸۰	● معرفی Font Dialog
۲۸۱	● معرفی File Dialog
۲۸۲	● معرفی Message Dialog
۲۸۳	● معرفی Qt Quick Layouts
۲۸۵	● معرفی Column Layout
۲۸۸	● معرفی Grid Layout
۲۹۹	● معرفی Row Layout
۲۹۴	● معرفی Stack Layout
۲۹۶	● معرفی Qt Quick Control Styles (سبک و استایل نویسی کنترل ها - سفارشی سازی)
۳۱۰	● واکنش گرایی و پاسخ دهی
۳۱۳	● محتواهی وب در اپلیکیشن با Qt WebEngine
۳۲۴	● محتواهی چند رسانه‌ای در کیوت QMultimedia
۳۲۵	● محتواهی چند رسانه‌ای در کیوت QMultimedia
۳۲۶	● پخش صوت
۳۲۷	● ضبط صدا در فایل
۳۲۸	● پخش ویدیو
۳۳۰	● کار با دوربین

## فصل هفتم

۳۳۲	● معرفی و پیکربندی کار با بانک اطلاعاتی (دیتابیس)
۳۳۶	● کار با بانک اطلاعاتی و ارتباط آن بین C++ و QML
۳۴۳	● معرفی و کار با XML

۳۴۷	..... معرفی و کار با JSON
۳۴۹	..... معرفی و کار با QSetting
۳۵۲	..... سفارشی سازی فایل pro. پروژه

## فصل هشتم

۳۵۵	..... مقایسه انواع حالت‌های کامپایل Debug و Release
۳۵۷	..... نحوه افزودن دیگر کتابخانه‌های C++ در محیط Qt Creator و استفاده همراه با کتابخانه Qt
۳۵۷	..... فرق بین کامپایل استاتیک و داینامیک
۳۵۸	..... نحوه خروجی گرفتن / گسترش (Deployment) در Qt
۳۵۸	• پیکربندی و انتشار برنامه در پلتفرم ویندوز (Windows)
۳۶۱	• پیکربندی و انتشار برنامه در پلتفرم مک (macOS)
۳۶۶	• پیکربندی و انتشار برنامه در پلتفرم لینوکس (Linux)
۳۶۸	• پیکربندی و انتشار برنامه در پلتفرم‌های iOS، iPad و iPhone
۳۸۱	• پیکربندی و انتشار برنامه در پلتفرم اندروید (Android)
۳۹۰	..... معرفی ابزار کیوبس (QBS)
۳۹۸	..... بهروزرسانی کیوت بدون دریافت فایل نصب آفلاین
۳۹۹	..... اهداف نسخه‌های کیوت و چشم‌انداز فنی کیوت ۶
۴۰۵	..... پیشنهادات و ملاحظات در عملکرد و کارآیی
۴۲۳	..... حق نشر کتاب و اهداف در نسخه‌ی بعدی کتاب
۴۲۴	..... تشکر و پیشنهاد از طرف نویسنده

## C++ مقدمه

### سی‌پلاس‌پلاس (C++) با آوای (سی‌پلاس‌پلاس) یکی از مدرن‌ترین و قدرتمندترین زبان‌های برنامه‌نویسی غالب و ساخت‌بشر تا به امروز

است که با هدف نوشتن برنامه‌های کارا و ساخت‌یافته و همچنین موضوعی یا (شیء‌گرا) برای برنامه‌نویسان است. این نام منسوب به ریک ماسکیتی (اواسط ۱۹۸۳) است و برای اولین بار در دسامبر سال ۱۹۸۳ به کار برده شده است. در طول مدت تحقیق این زبان بنام «C جدید» و بعدها «با کلاس» خوانده شد. در علوم کامپیوتر هنوز هم C++ به عنوان ابرساختار C شناخته می‌شود. آخرین نام از عملگر ++ در زبان C که برای افزایش مقدار متغیر به اندازه یک واحد بکار می‌رود و یک عرف معمول برای نشان دادن افزایش قابلیت‌ها توسط + ناشی گشته است. با توجه به نقل قولی از استراتژی سازنده سی‌پلاس‌پلاس : «این نام ویژگی‌ها تکاملی زبان در C را نشان می‌دهد.» C+ نام زبانی غیرمرتبه به این زبان است.

#### برخی از قابلیت‌ها

این زبان دارای قابلیت‌های انواع داده‌ایستا، نوشتار آزاد، چندملی، معمولاً زبان ترجمه شده با پشتیبانی از برنامه‌نویسی ساخت‌یافته، برنامه‌نویسی شیء‌گرا، برنامه‌نویسی جنریک است. C++ به همراه جد خود C از پرطرفدارترین زبان‌های برنامه‌نویسی تجاری هستند بنابراین در زیر فلسفه‌ای از این زبان را بیان می‌کنیم:

- زبان C++ طراحی شده است تا یک زبان عمومی با کنترل نوع ایستا و همانند C قابل حمل و پردازده باشد.
- زبان C++ طراحی شده است تا مستقیماً بصورت جامع از چندین شیوه برنامه‌نویسی (ساخت‌یافته، شیء‌گرا، انتزاع داده، و جنریک) پشتیبانی کند.
- زبان C++ طراحی شده است تا به برنامه‌نویس امکان انتخاب دهد حتی اگر این انتخاب اشتباه باشد.
- زبان C++ طراحی شده است تا حداکثر تطابق با C وجود داشته باشد و یک انتقال راحت از C را ممکن سازد.
- زبان C++ از به کاربردن ویژگی‌های خاص که مانع از عمومی شدن است خودداری می‌نماید.
- زبان C++ از ویژگی‌هایی که بکار برده نمی‌شوند استفاده نمی‌کند.
- زبان C++ طراحی شده است تا بدون یک محیط پیچیده عمل نماید.

#### ساختار برنامه در این زبان چگونه است؟

هر برنامه معمولاً از تعداد زیادی فایل تشکیل می‌شود که به هم الحق می‌گردند (با دستور include) و به این فایل‌های الحقی سرآیند (Header) می‌گوییم. فایل‌های الحقی کدها یا نسخه‌های اجرایی کلاس‌ها (مجموعه متغیرها و توابع) می‌باشند که در بدنه اصلی برنامه از آنها استفاده می‌شود. معمولاً هر کلاس (که تعریف یک نوع داده‌ای با متدهای مربوط به آن است) را در یک سرآیند می‌نویسند. هر سرآیند که معمولاً تنها تعاریف (معرفی) کلاس را در خود دارد به همراه فایل‌های پیاده سازی به زبان C++ یا پیاده سازی‌های کامپایل شده است (به صورت فایل اشیاء مانند dll یا so ... ) می‌تواند به کار برده شود. به مجموعه‌های یکپارچه‌ای از کلاس‌های پیاده سازی شده (به صورت فایل‌های سرآیند با پیاده سازی‌های کد یا اشیای زبان ماشین) که برای برنامه‌نویسی به کار می‌روند، یک کتابخانه C++ گفته می‌شود و یکی از ویژگی‌های پر طرفدار این زبان امکان به کارگیری کتابخانه‌های آماده و از پیش نوشته شده توسط دیگران است.

## کتابخانه‌ها در این زبان چگونه هستند؟

کتابخانه‌های بزرگ C++ مانند Boost, Qt, MFC, STL, wxWidgets و ... مجموعه قدرتمندی برای تولید برنامه در این زبان ایجاد کرده‌اند که هر کدام با منظور و هدف مشخصی ساخته شده‌اند.

## فرق بین C و C++ چیست؟

زبان C زبان برنامه‌نویسی Structure (ساخت‌یافته) است اما C++ زبان برنامه‌نویسی Object Oriented (شیء‌گرا) می‌باشد و از لحاظ دستوری فرق زیادی با هم ندارند، اما تفاوت عمدی این است که از شیء‌گرایی پشتیبانی می‌کند که سی C این قابلیت را ندارد.

## ویژگی‌های معرفی شده در C++ چیست؟

در مقایسه با C زبان C++ ویژگی‌های جدیدی را معرفی نموده است مانند تعریف متغیر به عنوان عبارت، تغییر نوع‌های همانند تابع، new حذف، نوع داده bool، توابع درون‌خطی، آرگومان پیشفرض، گرانبارسازی عملگر و تابع، فضای نام و عملگر تعیین حوزه ::، کلاس‌ها (شامل تمام ویژگی‌های مربوط به کلاس‌ها همانند وراثت، اعضای تابع، توابع مجازی، کلاس‌های انتزاعی، و سازنده‌ها)، قالب‌ها، پردازش استثناء، کنترل نوع زمان اجرا، عملگرهای سربار شده ورودی (<>) و خروجی (<>). برخلاف باور عموم C++ نوع داده ثابت را معرفی ننموده است. کلمه const کمی پیش از استفاده از این کلمه در C++ توسط زبان C بصورت رسمی بکار گرفته شد و در بعضی حالات C++ تعداد کنترل نوع بیشتری نسبت به زبان C انجام می‌دهد. بعضی ویژگی‌های C++ بعداً توسط C به کار گرفته شد مانند نحوه تعریف for، توضیحات به شکل (++) با استفاده از //، و کلمه inline با وجود اینکه تعریف این کلمه در C با تعریف آن در زبان C++ هماهنگی ندارد. همچنین در C ویژگی‌هایی معرفی شده است که در C++ وجود ندارند مانند ماکروهای قابل تغییر و استفاده بهتر از آرایه‌ها به عنوان آرگومان. بعضی کامپایلرها این ویژگی‌ها را پیاده نموده‌اند اما در بقیه‌این ویژگی‌ها موجب ناهماهنگی می‌گردد.

## ویژگی‌های و وضعیت نسخه‌های ۱۱، ۱۴، ۱۷ و ۲۰ چگونه است؟

زبان C++11 (معروف به C++ 0x) یک نسخه استاندارد از زبان C++ است که در ۱۲ آگوست ۲۰۱۱ منتشر و توسط ISO جایگزین C++ 03 شد این نسخه دارای نشان ISO/IEC 14882:2011 است و در تاریخ ۱۸ آگوست ۲۰۱۴ نسخه جدید آن یعنی C++14 منتشر و جایگزین C++11 شد. ۱۴ C++ روی اشکال‌زدایی و بهبودهای جزئی استاندارد قبلی یعنی C++11 تمرکز کرده است؛ این زبان در تاریخ ۱۵ می ۲۰۱۳ منتشر و در ۱۵ آگوست ۲۰۱۴ بعد از رای‌گیری و انجام تغییراتی جزئی استاندارد این زبان منتشر شد. بدلیل این که عموماً تاریخ انتشار این زبان بطور قابل ملاحظه‌ای دیر هنگام بوده است به C++14 گاهی نیز گفته می‌شود. همانند استاندارد C++11 که به آن C++0x گفته می‌شده و قرار بر این بوده که قبل از ۲۰۱۰ منتشر شود (البته تا سال ۲۰۱۱ انتشار به توعیق افتاد). در C++11 و C++14 ت و توابع جدیدی به هسته اصلی زبان و کتابخانه استاندارد آن اضافه شده است که شامل بسیاری از کتابخانه‌های C++TR1 به استثنای کتابخانه توابع ریاضی ویژه است. نسخه بعدی استاندارد این زبان با نام غیر رسمی C++17 و رسمی (C++1z) برای سال ۲۰۱۷ منتشر شده است. در حالی که C++ به هیچ مؤسسه‌ای وابسته نیست این مستندات به صورت آزادانه در دسترس نیستند. اما این نسخه از زبان همراه با کامپایلر GCC7 در اواسط سال ۲۰۱۷ در اختیار عموم قرار گرفت و در حال حاضر، با توجه به زمانی که این کتاب تألیف شده است، استاندارد 2a یا همان نسخه ۲۰ زبان برای سال ۲۰۲۰ برنامه‌ریزی می‌شود.

مواردی که در نسخه ۱۱ تحت کمیته استاندارد سازی شده است به صورت زیر هستند:

- حفظ پایداری و سازگاری با استاندارد C++ و تا حد امکان زبان C
- ترجیح دادن اضافه کردن ویژگی‌های جدید به کتابخانه استاندارد به توسعه هسته اصلی زبان
- ترجیح دادن تغییراتی که به تکنیک‌های برنامه‌نویسی کمک کند
- بهبود زبان برای راحت‌تر شدن توسعه کتابخانه و نه صرفاً اضافه کردن امکاناتی برای کاربردهای خاص
- بهبود اینمنی نوع با فراهم کردن جایگزین‌های امن تر برای روش‌های قدیمی
- بهبود سرعت اجرا و امکان کار مستقیم با سخت‌افزار
- فراهم کردن گزینه‌های مناسب برای مشکلات دنیای واقعی
- اجرای مفهوم «نداشتن سربارگذاری بی‌مورد» (پشتیبانی از موارد خاص فقط زمان استفاده از ابزارهای خاص لازم باشد برای نمونه تا زمانی که از std::thread استفاده نکردید نیازی به پشتیبانی سیستم‌عامل از برنامه‌نویسی موازی ندارید)
- تبدیل سی++ به زبانی که هم برای تدریس ساده باشد و هم برای یادگیری

مواردی که در نسخه ۱۴ تحت کمیته استاندارد سازی شده است به صورت زیر هستند:

- تشخیص نوع بازگشتی از تابع (Function return type deduction)

تشخیص نوع auto

بهبود در کلمه‌کلیدی constexpr

قالب‌های متغیر (variable templates)

مقدار دهنی انبوه به عضوهای کلاس (Aggregate member initialization)

الفاظ دودویی (Binary literals)

جدا کنند ارقام (Digit operators)

لامبدهای جنریک (generic lambdas)

پذیرش عبارت در گیرنده لامبدها (Lambda captures expression)

اضافه شدن صفت (Lambda captures expression)

مواردی که در نسخه ۱۷ تحت کمیته استاندارد سازی شده است به صورت زیر هستند:

- ساخت پیغام متنی برای static\_assert
  - اجازه استفاده typename به عنوان یکی دیگر از پارامترهای قالب Template
  - قوانین جدید برای auto
  - فضای نام‌های تودرتو و تعاریف دیگر مانند {...}::Y به جای {{...}}::X
  - اجازه دادن خواص‌ها برای فضاهای نام و شمارنده‌ها
  - خواص‌های جدید استاندارد [[nodiscard]], [[maybe\_unused]]، [[fallthrough]]
  - کاراکترهای UTF-8 تحت اللفظی
  - ثبات تکامل برای همه non-type ها در پارامترهای قالب Template
  - عبارت شکننده، برای قالب‌های واریادیک
  - دستور if برای زمان کامپایل در قالب of constexpr (expression)
  - ساختار اتصال به اعلان‌ها فراهم می‌کند: auto [a, b] = getTwoReturnValues();
  - مقدار دهی در عبارات if و switch
  - تضمین کپی نامفهوم توسط کامپایلر در بعضی از موارد
  - در اختصاص حافظه برخی از اضافات هم تراز شده‌اند
  - بهبود قالب std::pair<double, bool>(5.0, false)، اجازه استفاده به صورت std::pair(5.0, false) به جای Templates در سازنده‌ها
  - متغیرهای inline، اجازه تعریف در فایل سرآیند را دارند
  - به طور مستقیم توسط پیش پردازنده‌ها توسط \_\_has\_include بررسی می‌شود که سرآیند مربوطه در دسترس است یا خیر
  - مقدار \_\_cplusplus به 201703L تغییر یافته است
  - انواع داده‌ای که تحت استاندارد ISO تایید شده‌اند
- std::string\_view    ○
- std::optional    ○
- std::any    ○
- std::variant    ○
- std::span    ○
- std::byte    ○
- std::invoke    ○
- std::apply    ○

	std::is_callable	○
	std::make_from_tuple	○
	std::is_callable	○
گزینه std::uncaught_exception جایگزین std::uncaught_exception ر شده است		●
تابع جدید درج کننده std::unordered_map برای insert_or_assing و try_emplace فراهم شده است		●
یکسان سازی دسترسی به ظروف std::empty ، std::size و std::data		●
امکان تعریف الفاظ رشته‌ای پیوسته		●
برخی از توابع منسخ شده مانند std::random_shuffle و std::auto_ptr حذف شده‌اند		●
ماژول فایل سیستم بر پایه boost::filesystem تعبیه شده است		●
الگوریتم‌های موازی از STL		●
افزوده شدن توابع ویژه ریاضیات، شامل انتگرال‌ها و توابع بسل		●
صفات عملگرهای منطقی مانند std::negation و std::disjunction ، std::conjunction		●
برخی از الگوریتم‌های جدید		●
for_each_n	○	
reduce	○	
transform_reduce	○	
exclusive_scan	○	
inclusive_scan	○	
transform_exclusive_scan	○	
transform_inclusive_scan	○	
در رابطه با چندنخی موارد زیر		●
std::shared_mutex	○	
atomic<T>::is_always_lockfree	○	
scoped_lock<Mutexes...>	○	
فایل سیستم‌ها		●
[class.path]	○	
[class.filesystem.error]	○	
[class.file_status]	○	
[class.directory_entry]	○	

[class.recursive\_directory\_iterator] و [class.directory\_iterator]

[fs.ops.funcs]

## کامپایلرهای C/C++ و وضعیت آن‌ها

مترجم، همگردان یا کامپایلر (Compiler) برنامه یا مجموعه‌ای از برنامه‌های کامپیوتی است که متنی از زبان برنامه‌نویسی سطح بالا (زبان مبدأ) را به زبانی سطح پایین (زبان مقصد)، مثل اسembly یا زبان سطح ماشین، تبدیل می‌کند. خروجی این برنامه ممکن است برای پردازش شدن توسط برنامه دیگری مثل پیوند دهنده مناسب باشد یا فایل متنی باشد که انسان نیز بتواند آن را بخواند. مهم‌ترین علت استفاده از ترجمه کد مبدأ، ایجاد برنامه اجرایی است. بر عکس برنامه‌ای که زبان برنامه‌نویسی سطح پایین را به بالاتر تبدیل می‌کند را مترجم وارون گوییم.

ترجمه کامل کد منبع برنامه‌ای از یک زبان سطح بالا به کد شیء، پیش از اجرای برنامه را همگردانی یا کامپایل می‌گویند.

به بیان ساده، کامپایلر برنامه‌ای است که یک برنامه نوشته شده در یک زبان خاص ساخت یافته را خوانده و آن را به یک برنامه مقصد (Target) تبدیل می‌نماید. در یکی از مهم‌ترین پروسه‌های این تبدیل، کامپایلر وجود خطا را در برنامه مبدأ اعلام می‌نماید. انواع کامپایلرهایی که از طرف شرکت‌های سازنده و توسعه‌دهنده ارائه شده‌اند مانند Clang، GCC، MSVC، Intel و غیره که آخرین نسخه‌های آن‌ها ویژگی‌های کامل C++ شماره ۱۱، ۱۴، ۱۷ و حتی ۲۰ را پشتیبانی می‌کنند.

## ویژگی‌های نسخه ۱۱ زبان، تحت کامپایلرهای ذکر شده

Intel C++	EDG ecpp	MSVC	Clang	GCC	ویژگی‌های نسخه ۱۱
15.0	4.8	19.0*	3.0	4.8	alignas
15.0	4.8	19.0*	2.9	4.5	alignof
13.0	Yes	17.0*	3.1	4.4	Atomic operations
11.0(v0.9) 12.0(v1.0)	4.1(v0.9)	16.0 (v0.9)*	بله	4.4(v1.0)	auto
11.1	4.1	19.0*	بله	4.3	C99 preprocessor
13.0* 14.0	4.6	19.0*	3.1	4.6	constexpr
11.0(v1.0) 12.0(v1.1)	4.1(v1.0)	16.0 (v1.1)*	2.9	4.3(v1.0) 4.8.1(v1.1)	decltype
12.0	4.1	18.0*	3.0	4.4	Defaulted and deleted functions
14.0	4.7	18.0*	3.0	4.7	Delegating constructors
13.0	4.4	18.0*	3.0	4.5	Explicit conversion operators
11.1* 12.0	4.1	16.0*	2.9	4.7	Extended friend declarations
9.0	3.9	12.0*	Yes	3.3	extern template
11.1* 14.0	4.5	17.0*	3.1	4.6	Forward enum declarations

15.0	4.8	19.0*	3.3	4.8	Inheriting constructors
13.0* 14.0	4.5	18.0*	3.1	4.4	Initializer lists
12.0(v1.1)	4.1(v1.1)	16.0*	3.1	4.5(v1.1)	Lambda expressions
12.0	4.2	16.0*	2.9	4.5	Local and unnamed types as template parameters
بله	بله	بله	بله	بله	long long
14.0	4.5	19.0*	2.9	4.4	Inline namespaces
12.1* 14.0	4.4	19.0*	2.9	4.4	New character types
12.0	4.1	16.0*	2.9	4.4	Trailing function return types
12.1	4.2	16.0*	2.9	4.6	nullptr
11.0*	4.7	19.0*	3.0	4.4	Unicode string literals
14.0	4.7	18.0*	Yes	4.5	Raw string literals
15.0	4.8	19.0*	3.1	4.7	User-defined literals
11.0	4.1	14.0*	بله	4.3	Right angle brackets
11.1(v1.0) 12.0(v2.0) 14.0(v3.0)	4.5(v3.0)	16.0*	بله	4.3(v1.0) 4.5(v2.1) 4.6(v3.0)	R-value references
11.0	4.1	16.0*	2.9	4.3	static_assert
13.0	4.0	17.0*	2.9	4.4	Strongly-typed enum
12.1	4.2	18.0*	3.0	4.7	Template aliases
11.1* 15.0*	4.8	19.0*	3.3* 3.3	4.4* 4.8	Thread-local storage
14.0*	4.6	19.0*	3.0	4.6	Unrestricted unions
10.0	4.0	14.0*	3.0	4.3	Type traits
12.1(v1.0)	4.1(v0.9)	18.0*	2.9(v1.0)	4.3(v0.9) 4.4(v1.0)	Variadic templates
13.0	4.5	17.0*	3.0	4.6	Range-for loop
12.0(v0.8) 14.0(v1.0)	4.8(v1.0)	17.0*	2.9	4.7	override and final
12.1	4.2	19.0*	3.3	4.8	Attributes
14.0	4.7	19.0*	2.9	4.8.1	ref-qualifiers
14.0	4.6	18.0*	3.0	4.7	Non-static data member initializers
11.1*	بله	19.0*	2.9	4.3	Dynamic initialization and destruction with concurrency (magic statics)
14.0	4.5	19.0*	3.0	4.6	noexcept
خیر	خیر	خیر	خیر	خیر	dynamic pointer safety (GC interface)

خیر	خیر	19.0*	3.8	5.1	Money, Time, and hexfloat I/O manipulators
-----	-----	-------	-----	-----	--

### ویژگی‌های نسخه ۱۴ زبان، تحت کامپایلرهای ذکر شده

Intel C++	EDG ecpp	MSVC	Clang	GCC	ویژگی‌های نسخه ۱۴
16.0	4.9	18.0*	3.4	4.9	Tweaked wording for contextual conversions
11.0	4.10	19.0*	2.9	4.3/4.9	Binary literals
15.0	4.9	19.0*	3.3/3.4	4.8/4.9	decltype(auto), Return type deduction for normal functions
15.0	4.10	19.0*	3.4	4.5/4.9	Initialized/Generalized lambda captures (init-capture)
16.0	4.10	19.0*	3.4	4.9	Generic (polymorphic) lambda expressions
17.0	4.11	19.0*	3.4	5.0	Variable templates
17.0	4.11	19.1*	3.4	5	Extended constexpr
16.0	4.9	19.1*	3.3	5	Member initializers and aggregates (NSDMI)
خیر	نامشخص	نامشخص	3.4	نامشخص	Clarifying memory allocation (avoiding/fusing allocations)
15.0* 16.0	4.9	19.0*	3.4	4.9	Deprecated attribute
17.0	4.10.1	19.0*	3.4	5	Sized deallocation
16.0	4.10	19.0*	3.4	4.9	Single quote as digit separator
نامشخص	نامشخص	19.0*	Yes	5.0	std::result_of and SFINAE
نامشخص	نامشخص	19.0*	3.4	5.0	constexpr for <complex>
نامشخص	نامشخص	19.0*	3.4	5.0	constexpr for <chrono>
نامشخص	نامشخص	19.0*	3.4	5.0	constexpr for <array>
نامشخص	نامشخص	19.0*	3.4	5.0	constexpr for <initializer_list>, <utility> and <tuple>

نامشخص	نامشخص	19.0*	3.4	5.0	Improved std::integral_constant
نامشخص	نامشخص	19.0*	3.4	5.0	User-defined literals for <chrono> and <string>
نامشخص	نامشخص	19.0*	3.4	5.0*	Null forward iterators
نامشخص	نامشخص	19.0*	3.4	5.0	std::quoted
نامشخص	نامشخص	19.0*	3.4	5.0	Heterogeneous associative lookup
نامشخص	نامشخص	19.0*	3.4	5.0	std::integer_sequence
نامشخص	نامشخص	19.0*	3.4	5.0	std::shared_timed_mutex
نامشخص	نامشخص	19.0*	3.4	5.0	std::exchange
نامشخص	نامشخص	19.0*	3.4	5.0	fixing constexpr member functions without const
نامشخص	نامشخص	19.0*	3.4	5.0	std::get<T>()
نامشخص	نامشخص	19.0*	3.4	5.0	Dual-Range std::equal, std::is_permutation, std::mismatch

### ویژگی‌های نسخه ۱۷ زبان، تحت کامپایلرهای ذکر شده

Intel C++	EDG ecpp	MSVC	Clang	GCC	ویژگی‌های نسخه ۱۷
17.0	4.10.1	19.0*	3.8	5.0	New auto rules for direct-list-initialization
18.0	4.12	19.1*	2.5	6	static_assert with no message
17.0	4.10.1	19.0*	3.5	5.0	typename in a template template parameter
خیر	خیر	10.0*	3.5	5.1	Removing trigraphs
17.0	4.12	19.0*	3.6	6	Nested namespace definition
خیر	4.11	19.0*	3.6	4.9 (namespaces) / 6 (enumerators)	Attributes for namespaces and enumerators
17.0	4.11	19.0*	3.6	6	u8 character literals
خیر	خیر	خیر	3.6	6	Allow constant evaluation for all non-type template arguments
خیر	خیر	خیر	3.6	6	Fold Expressions

خیر	خیر	خیر	3.8	7	Remove Deprecated Use of the register Keyword
خیر	خیر	خیر	3.8	7	Remove Deprecated operator++(bool)
خیر	خیر	خیر	4	7	Removing Deprecated Exception Specifications from C++17
خیر	خیر	خیر	4	7	Make exception specifications part of the type system
خیر	خیر	خیر	3.9	7	Aggregate initialization of classes with base classes
خیر	خیر	خیر	3.9	7	Lambda capture of *this
خیر	خیر	خیر	3.9	7	Using attribute namespaces without repetition
خیر	خیر	خیر	4	7	Dynamic memory allocation for over-aligned data
خیر	خیر	خیر	3.9	6	Unary fold expressions and empty parameter packs
خیر	خیر	خیر	Yes	5.0	_has_include in preprocessor conditionals
خیر	خیر	خیر	5	7	Template argument deduction for class templates
خیر	خیر	خیر	4	7	Non-type template parameters with auto type
خیر	خیر	خیر	4	7	Guaranteed copy elision
خیر	خیر	خیر	3.9	7	New specification for inheriting constructors (DR1941 et al)
خیر	خیر	خیر	3.9	7	Direct-list-initialization of enumerations
خیر	خیر	خیر	4	7	Stricter expression evaluation order
خیر	خیر	خیر	5	7	constexpr lambda expressions

خیر	4.12	19.1*	3.9	6	Differing begin and end types in range-based for
خیر	خیر	19.1*	3.9	7	[[fallthrough]] attribute
خیر	خیر	خیر	3.9	7	[[nodiscard]] attribute
خیر	خیر	خیر	4	7	Pack expansions in using-declarations
خیر	خیر	خیر	3.9	7	[[maybe_unused]] attribute
خیر	خیر	خیر	4	7	Structured Bindings
خیر	خیر	خیر	Yes	3.0	Hexadecimal floating-point literals
خیر	خیر	خیر	3.9	Yes	Ignore unknown attributes
خیر	خیر	خیر	3.9	7	constexpr if statements
خیر	خیر	خیر	3.9	7	init-statements for if and switch
خیر	خیر	خیر	3.9	7	Inline variables
خیر	خیر	خیر	4	7	DR: Matching of template template-arguments excludes compatible templates
خیر	خیر	19.0*	3.7	6	std::uncaught_exceptions()
خیر	خیر	خیر	خیر	7	Splicing Maps and Sets
نامشخص	نامشخص	19.0*	4	بله	Improving std::pair and std::tuple
نامشخص	نامشخص	19.0*	3.7	6	std::shared_mutex(untimed)
خیر	خیر	خیر	خیر	خیر	Elementary string conversions
خیر	خیر	خیر	خیر	خیر	std::string_view

لیست کامل انواع کامپایلرها: [https://en.wikipedia.org/wiki/List\\_of\\_compilers](https://en.wikipedia.org/wiki/List_of_compilers)

## ساختار اسناد در این زبان چگونه است؟

در رابطه با ساختار برنامه‌های نوشته شده توسط C++ بدانید که منظور از ساختار در اینجا انواع فایل‌های موجود در زبان C++ است، در این رابطه باید اینگونه اشاره کرد که در این زبان می‌توانیم از فایل‌های زیر برای برنامه‌نویسی استفاده کنیم.

- فایل با پسوند (.c) این فایل منبعی برای کدهایی از نوع زبان C هستند.
- فایل با پسوند (c++) از نوع زبان C و C++ هستند ضعف این نوع فایل در قابل حمل نبودن و عدم شناسایی توسط فایل سیستم‌ها می‌باشد.
- فایل با پسوند (cxx) منبعی برای کدهایی از نوع زبان C و C++ هستند با تفاوت که نسبت به فایل c++.c قابل حمل‌تر است.
- فایل با پسوند (cpp) منبعی برای کدهایی از نوع زبان C و C++ هستند یعنی در هر دو نیز قابل استفاده است. این پسوند با تمامی سیستم‌ها سازگاری دارد و بسیار رایج است.
- فایل با پسوند (hxx) معمولاً فایل با عنوان (سرآیند) یاد می‌شوند و معمولاً فقط حاوی اعلان‌ها است.
- فایل با پسوند (hpp) معمولاً فایل با عنوان (سرآیند) یاد می‌شوند و معمولاً فقط حاوی اعلان‌ها است. این قالب توسط مارس دیجیتال استفاده می‌شود. همچنین بورلند و دیگر کامپایلرهای سی++ از آن پشتیبانی می‌کنند. ممکن است در این فایل متغیرها، ثوابت و توابع که در فایل منبع اصلی به آن‌ها اشاره شده است اعلام شود.
- فایل با پسوند (h) معمولاً فایل با عنوان (سرآیند) یاد می‌شوند و معمولاً فقط حاوی اعلان‌ها است این نوع بسیار رایج است و تقریباً با تمامی سیستم‌ها سازگاری دارد. (پشنهداد شده)
- فایل با پسوند (hh) در این زبان : فایل با عنوان (سرآیند) یاد می‌شوند و معمولاً فقط حاوی اعلان‌ها است.
- فایل با پسوند (h++) در این زبان : این نوع فایل‌ها معمولاً فایل با عنوان (سرآیند) یاد می‌شوند و معمولاً فقط حاوی اعلان‌ها است. ضعف این نوع فایل در قابل حمل نبودن و عدم شناسایی توسط فایل سیستم‌ها می‌باشد.
- نکته: معمولاً روش استفاده از این فایل‌ها به صورت جفت می‌باشد. برای مثال استفاده از فایل سرآیند h.++ همراه با سورس c.++ فراهم می‌شود. فایل .cxx همراه با .hh و فایل‌های .cpp و .hpp و در نهایت فایل .c و .h باهم دیگر مورد استفاده قرار می‌گیرند. اما در این میان استاندارد ترین و پر کاربرد ترین آن‌ها که از لحاظ قابل حمل بودن هم از طرف کمیته استاندارد سازی ISO CPP قابل تایید هستند فایل‌های .cpp و .c و .h و .hpp می‌باشند.

یک فایل سرآیند با پسوند (hpp, h, ...) می‌تواند شامل محتوای زیر باشد:

- تعریف کلاس همراه با اعضای آن:

```
#ifndef CPPFILES_H
#define CPPFILES_H

extern int status;

class CPPfiles
```

```
{
public:
    CPPFiles();
public:

    void myFunction();
    int safe(int i);
    inline int enable = false;
};
```

```
#endif // CPPFILES_H
```

یک فایل منبع - سورس با پسوند (.cxx, .c, .hpp...) می‌تواند شامل محتوای زیر باشد:

- تعریف کلاس، تعریف توابع، اعلام شیء

```
#include "cppfiles.h"

int status = 1;

CPPFiles::CPPFiles() { enable = true }

void CPPFiles::myFunction() {

    //Do somthing...
}

int CPPFiles::safe(/*@Param*/)

{
    return /*Somthing...*/;
}
```

انواع فایل‌هایی که به آنها اشاره شده است بسیار هستند ولی متناسب با محبوبیت و پشتیبانی کامپایلر (هم‌گردان‌ها) از این فایل‌ها در این زبان برای انتخاب آنها مهم است بنابراین در طی آموزش و تمامی مراحل ما فقط از فایل‌های h. برای سرآیند و فایل‌های cpp. برای منابع استفاده خواهیم کرد.

### چرا و چه زمانی باید از فایل‌های .h و چه زمانی از فایل‌های .cpp استفاده کنیم؟

در پاسخ باید اینگونه اشاره کرد که در زبان C++ برای راحتی کار به خاطر پیچیده بودن کدنویسی امکان این وجود دارد که توابع، کلاس‌ها و تعاریف را در فایل جداگانه‌ای مانند سرآیند فایل یا همان h. بنویسیم و برای اجرا عملیات و فراخوانی از فایل‌های cpp. استفاده کنیم. البته این اجراء نیست شما می‌توانید در فایل h. و یا فایل cpp. به تنهایی برنامه را اجرا کنید که معمولاً در مثال‌های کوچک و کوتاه از یک نوع فایل استفاده می‌شود.

### کاربرد این زبان در کجاست؟

معمولًاً تمامی برنامه‌ها و نرم‌افزارهایی که به صورت روزمره در زندگی مدرن امروزی مشاهده می‌کنیم بدون شک توسط زبان غالبی مانند سی++ نوشته شده‌اند. به عنوان مثال انواع صنایع در کشورها از قبیل صنعت خودرو سازی، صنعت فضایی، سیستم‌های معماری و بانکی، تجهیزات مدرن و سخت‌افزارها، رباتیک در انواع صنایع، سیستم‌های کامپیوتری و یا کنسول‌های بازی، سیستم‌های خانگی و یا سیستم‌های رباتیک و هوش مصنوعی، تجهیزات مجهر به انواع حسگرها و هوش مصنوعی در علم پزشکی، صنعت بازی سازی، نرم‌افزارهای مهندسی و همچنین سیستم‌عامل‌ها

و بسیاری از موارد دیگری که می‌توان نام برد بدون شک توسط این زبان پیاده سازی شده‌اند. بهتر است به این نکته نیز توجه داشته باشید که سی++ به عنوان یک زبان غالب همراه چهار زبان مشخص برای فناوری بلاک‌چین نیز در آینده‌ای از فناوری بسیار طرفدار دارد.

## کاربرد این زبان در زمینه وب چگونه است؟

معمولاً در وب زبان‌هایی مانند JavaScript کاربردهای بسیاری را دارد، اما سی‌پلاس‌پلاس نیز از این حوزه غافل نبوده و می‌توان توسط این زبان اقدام به ساخت و توسعه پروژه‌های بسیار قدرتمند کرد. به عنوان مثال هستهٔ بزرگترین وب‌سایتها مانند فیسبوک، گوگل و غیره با سی++ توسعه داده شده است.

همچنین برای اینکه به صورت فردی یا اهداف مشخص طراحی وب را تحت سی++ انجام دهید، لازم است کیوت و کتابخانهٔ پیش‌فرض این زبان را به خوبی بآموزید. همچنین کتابخانه‌هایی مانند Wt و Cutelyst یکی از بهترین کتابخانه‌های ساخت و توسعهٔ صفحات وب تحت سی++ می‌باشند که نتیجهٔ خروجی یک وب‌سایت به صورت بسیار چشم‌گیری متفاوت و شگفت‌انگیز است.

**طراحی رابط کاربری (UI) و تجربه کاربری (UX) در C++ چگونه است؟** برای طراحی رابط گرافیکی ابتدا باید ذهن خود را از محیط‌های توسعه انحصاری در پلتفرم‌های خاص مانند Visual Studio و Xcode دور سازید سپس با در نظر گرفتن برخی از کتابخانه‌ها اقدام به پیاده سازی رابط‌کاربری کنید که مناسب‌ترین و مدرن‌ترین آن‌ها Qt خواهد بود:

FLTK  
nana  
WxWidgets  
OWLNext  
GTK+  
webkitgtk  
flowcanvas  
evince  
Qt

## تولید اپلیکیشن در موبایل

همانطور که می‌دانید امروزه صنعت توسعه نرم‌افزار در بازار کار مرتبط با تبلت‌ها و موبایل یکی از پر درآمدترین صنف‌های توسعه محصول برای استارت‌آپ‌ها و کسب‌وکارهای موفق است و با سرعت بالایی رو به پیشرفت می‌باشد. بنابراین در این میان فناوری Cross Platform بودن نیز مطرح است، اگر نیاز است برنامه‌ای را تولید کنیم که در Android قابل اجرا باشد باید توسط زبان‌هایی مانند Java آن را توسعه دهیم و یا اگر نیاز است برای iOS برنامه‌ای را منتشر کنیم نیاز است با زبان Objective-C آشنا باشیم و همچنین محیط Xcode که مختص بسترهای Apple است و شما بدون داشتن سیستم Apple تحت سیستم‌عامل مک نخواهید توانست حتی پروژه خود را ایجاد و طراحی نمایید.

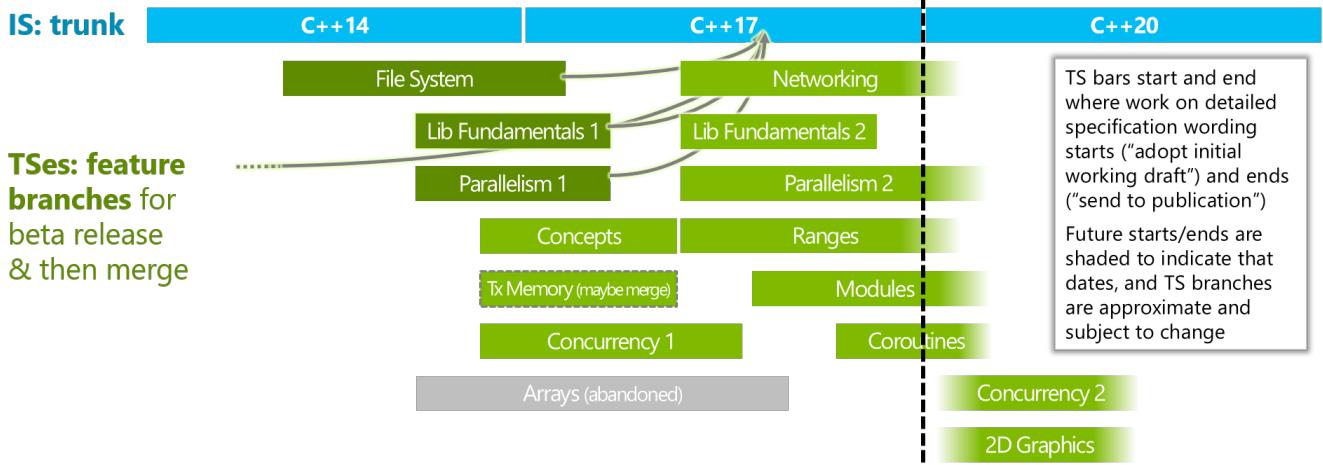
با توجه به اهمیت این موضوع، در بحث توسعه برنامه‌های موبایل محدودیت‌هایی وجود دارد که در هر ابزاری نمی‌تواند آن را در سطح یک ابزار بومی فراهم ساخت. بنابراین، زبان سی‌پلاس‌پلاس تحت کتابخانه‌های عظیمی که دارد برای هر یک از مشکلات و نیازها راه حل بسیار خوبی را ارائه می‌کند. البته فراموش نکنید که سی‌پلاس‌پلاس ذاتاً چند منظوره و یک زبان بومی در تمامی پلتفرم‌ها محسوب می‌شود و برای تولید و توسعه محصولات چند منظوره کافی است با ویژگی‌های این زبان و پلتفرم‌های هدف آشنایی کامل داشته باشیم. بدون شک این زبان بسیار قدرتمند است اما برای اینکه در بحث موبایل موفق باشید حتماً باید یک برنامه نویس متوجه به بالا در C++ باشید تا در زمان نیاز کدهای خود را به خوبی توسعه داده و آن را متناسب با پلتفرم مقصد (هدف) هماهنگ سازید.

## استانداردهای زبان

استاندارد سازی C++ توسط یک گروه به عنوان یک کمیته از تشکیلات ISO انجام می‌شود. تاکنون ۵ نسخه از استاندارد این زبان منتشر شده است؛ و استاندارد C++17 نیز برای انتشار در سال ۲۰۱۷ برنامه‌ریزی شده است.

وضعیت	نام غیررسمی	نام رسمی (علمی)	استاندارد C++	سال
منسوخ شده	C++98	C++98	ISO/IEC 14882:1998	۱۹۹۸
منسوخ شده	C++03	C++03	ISO/IEC 14882:2003	۲۰۰۳
منسوخ شده	C++07/TR1	C++07	ISO/IEC TR 19768:2007	۲۰۰۷
پشتیبانی شده	C++11	C++03	ISO/IEC 14882:2011	۲۰۱۱
پیشنهادی	C++14	C++1y	ISO/IEC 14882:2014	۲۰۱۴
منتشر شده	C++17	C++1z	ISO/IEC 14882:2017	۲۰۱۷
در حال توسعه و نهایی سازی	C++20	C++2a	هنوز تعیین نشده	۲۰۲۰

در زیر مراحل توسعه C++ مشخص شده است که در بین ۱۹۸۹ از نسخه ۹۸ تا ۲۰۱۹ و ۲۰۲۰ که برای نسخه ۲۰ مشخص شده است شده است. همچنین خلاصه‌ای از ویژگی‌های تأیید شده نیز آورده شده است که با یک نگاه سریع می‌توان آن‌ها را مشاهده کرد. تمامی اطلاعات و ویژگی‌های مرتبط با زبان برنامه‌نویسی مدرن سی++ در قالب بسته آموزشی ویدیو و کتاب اختصاصی خود ارائه می‌شود.



## مقدمه کیوت (Qt)

**کیوت (Qt)** مجموعه‌ای از کتابخانه‌ها و سرآیندهای نوشته شده به زبان C++ است که به برنامه‌نویس امکان توسعه آسان نرم‌افزارهای کاربردی

را می‌دهد. اگر به فکر این هستید که در این زمینه یک حرفه‌ای محسوب شوید بهتر است به جزئیات آن دقیق‌تر کنید! طبق تجربیاتی که من در نسخه قبلی کتاب داشته‌ام روشن است که برخی از علاقه‌مندان که رغبت چندانی به رعایت جزئیات و قوانین برنامه‌نویسی ندارند نام این کتابخانه را چه در حالت آوازی و چه در حالت نوشتاری به اشتباه بیان می‌کنند. بنابراین جهت اصلاح و اشاره دقیق به آن که شاید از نظر خواننده مهم تلقی نشود به نکاتی اشاره می‌شود.

نام کیوت در لاتین به صورت Qt نوشته می‌شود و حرف دوم آن یعنی t می‌بایست به صورت lowercase یا همان حرف کوچک نوشته شود. طبق این قانون شکل صحیح آن Qt است نه QT. و اما در رابطه با تلفظ این نام برخلاف انتظار آوای صحیح آن به صورت "کیوت" مشابه تلفظ "Cute" است و تلفظ کیوتی اشتباه است.

بنابراین توجه داشته باشید کیوت یک زبان نیست! لذا در تمامی مراحل تولید یک محصول شما به زبان C++ در بک‌اند و QML در فرانت‌اend در حال کد نویسی خواهد بود. همچنین کیوت شامل کلاس‌هایی برای کار با واسط گرافیکی، چندرسانه، ابزارهای پایگاه داده، شبکه و ... است. کیوت

همانند دیگر کتابخانه‌های سی‌پلاس‌پلاس بوده و یک کتابخانه از نوع غیر **STL** محسوب می‌شود که همراه هسته زبان ارائه نشده است. نرم‌افزارهای نوشته شده با ابزار کیوت قادرند تا با استفاده از یک کامپایلر (هم‌گردان) زبان **C++** برای طیف وسیعی از سیستم‌عامل‌ها از جمله گنو/لینوکس (نسخه‌های رومیزی و وسیله‌های قابل حمل)، ویندوز، ویندوز CE، مک او اس، آی او اس، اندروید و ... همگردانی شوند. بدین ترتیب حمل نرم‌افزار نوشته شده بدون تغییر در متن کد نوشته شده امکان‌پذیر است که همانند کتابخانه‌های دیگر **C++** مانند **STL**, **Poco**, **Boost**, **wxWidgets** و ... مورد استفاده قرار می‌گیرد با تفاوت اینکه در زمینه طراحی رابط گرافیکی نسبت به کتابخانه‌های دیگر پایدارتر و جامع‌تر است.

امروزه توسعه نرم‌افزار و بهروزرسانی‌های آن در انواع پلتفرم‌ها از قبیل Windows, Linux, macOS و همچنین پلتفرم‌های موبایل و تبلت از قبیل Backberry, iOS, Andoird و ... با سرعت بسیار زیادی دنبال می‌شود. همچنین آرزوی اکثر برنامه‌نویسان است که یک زبان ویژه با تمامی قابلیت‌ها و مهمتر از همه پشتیبانی از Objective Oriented و Performance (کارایی) بالا را همراه با یک IDE (محیط توسعه) همه‌کاره و جذاب در اختیار داشته باشند که در این صورت به جای تجربه کردن تمامی محیط‌های برنامه‌نویسی در این زمینه‌ها پیشنهاد می‌کنیم خیلی راه دوری نزولید زیرا با استفاده از محیط برنامه‌نویسی Qt که توسط زبان قدرتمند C++ مورد استفاده قرار می‌گیرد تقریباً همه آرزوهای شما در برنامه‌نویسی فراهم خواهد گردید.

## چرا باید از کیوت استفاده کنیم؟

زبان سی‌پلاس‌پلاس در حال توسعه و رشد بسیار زیادی است که بر اساس آخرین نظریه توسعه‌دهندگان بیش از ۹۵ درصد برنامه‌نویسی مدرن در حوزه اینترنت اشیاء تا سال ۲۰۲۰ نزدیک به ۳۰ میلیارد دستگاه را تحت سلطه کتابخانه Qt توسعه داده خواهند شد بنابراین طبیعی است که همانند زبان‌های دیگری در طی این سالها پیشرفت کرده‌اند زبان C++ هم خالی از پیشرفت نباشد و نسبت به قبل بسیار توانمند و قدرتمند شده است و در این میان نه تنها در رابطه با قابلیت‌ها موارد زیادی در نسخه‌های ۱۱ و ۱۴ و ۱۷ این زبان رفع و توسعه داده شده است در کنار این پیشرفت‌ها محیط توسعه بسیار جذابی به کمک برنامه‌نویسان و مشتاقان این زبان آمده است آن چیزی نیست به جز **Qt Creator** یک محیط توسعه بسیار قدرتمند و کامل؛

همه چیز ساده تر، روانتر و جذاب‌تر شده است و سرعت برنامه‌نویسی و طراحی فرم‌ها و قالب‌بندی‌های پیشرفته که قبلاً نیاز به نوشتن کدهای بیشتری داشت، بسیار بهتر از قبل شده است. به طوری که به جرأت می‌توان گفت در نگاه اول کار با Qt را می‌توان پسندید! این محیط برخلاف محیط‌های معروف دیگری مانند Visual Studio و Xcode به هیچ عنوان سیاست انصصاری بودن را ندارد و تنها بر روی سکوی ویندوز یا مک محدود نشده است که از قدرتمندترین IDE این دوره پشتیبانی می‌کند! بلکه سیستم‌عامل‌های قدرتمند دیگری بر پایه ایستگاه‌های یونیکس و لینوکس مانند Ubuntu و Linux macOS این محیط برنامه‌نویسی خارق العاده را کاملاً پشتیبانی می‌کند و این در ابتدای کار به تنها‌ی ارزشمند است.

از قابلیت‌هایی که نمی‌شود به این راحتی از آن‌ها چشم پوشی کرد می‌توان به قابلیت **Cross-platform** بودن برنامه‌های تولید شده توسط C++/Qt اشاره کرد که شما به راحتی می‌توانید خروجی را در سیستم‌عامل مورد نظرتان دریافت و کامپایل کنید. به جای درگیر شدن با محیط‌های توسعه Visual Studio, Android Studio, Xcode و ... کافی است محیط توسعه Qt Creator را در اختیار داشته باشید تا بتوانید برای هر آنچه که در ذهن دارید برنامه‌های خود را طراحی، توسعه و تولید نمایید.

نکته: کیوت جامع‌ترین و تنها ابزار چند-سکویی واقعی و بومی است!

اگر شما خواهنان برنامه‌نویسی مدرنی هستید که برای شما لذت بخش باشد بدون شک سی‌پلاس‌پلاس می‌تواند شما را شیفته خود کند. چرا که کتابخانه‌های بزرگ و قدرتمند این زبان همچون کیوت با تمام قدرت آمده‌اند تا شما را به یک برنامه‌نویس افسانه‌ای تبدیل کند. برنامه‌نویسی که بتواند با یک تیپ چندنشان را هدف قرار دهد. اگر در سر دارید یا یکی از آرزوهای شما این است که بتوانید به عنوان چندین برنامه‌نویس خود به تنها‌یی همهٔ کارها را انجام دهید این رویا توسط کیوت به حقیقت تبدیل خواهد شد! نیازی نیست شما نگران تولید محصول خود با محدودیت پلتفرم یا فناوری‌ها باشید. کافی است خلاقانه فکر کرده و محصول خود را بعد از اعتبار سنجی وارد مرحله توسعه نمایید. البته همهٔ این حرف‌ها به این معنی نیست که اگر شما تنها یک برنامه‌نویس بکاند یا فراتر از هستید خواهید توانست به راحتی برنامه‌نویسی کنید، چرا که برای حرف‌های شدن باید به مباحث کامل یک پلتفرم و زبان سی++ مسلط باشید.

یکی از بزرگترین مشکلاتی که در صنعت نرم‌افزاری و همچنین عدم موفقیت‌های کسب‌وکارهای نو پا مطرح می‌شود، درگیر بودن برنامه‌نویسان با ابزارها و زبان‌های برنامه‌نویسی متعدد است. برای مثال عاملین کسب‌وکارهای نو پا معمولاً به دنبال انواع برنامه‌نویس‌های متخصص در حوزه‌های ویندوز، لینوکس، مک، اندروید و غیره... می‌باشند که طبیعتاً هزینه‌ها و برنامه‌ریزی‌های زمانی خاص خودشان را می‌طلبند. این مشکل را می‌توان با در نظر گرفتن فناوری چند-سکویی تحت یک زبان قدرتمند و یک سازنده رابط سریع به بهترین حالت ممکن حل کرد. البته توجه داشته باشید برنامه‌نویسی چند-سکویی به صورت بومی تحت C++ قابل مقایسه با برنامه‌نویسی چندسکویی رایج تحت ابزارهای دیگر مانند فلاتر (Flutter) و یا Xamarin نیست.

نکته بسیار مهم که در این کتاب رعایت شده است

اصطلاحات موجود در برنامه‌نویسی بهتر است مستقیماً ترجمه نشوند، برای مثال در صورتی که Qt Quick را به کیوت سریع ترجمه شود اشتباه است. بنابراین در این کتاب سعی شده است تا در صورت امکان معنا را همراه با تلفظ و آواز صحیح بیان کنیم و اکثرًا اصطلاحات موجود به صورت اسم خاص عنوان خواهند شد که نمونه‌های آن مانند Qt Quick و غیره... است! بنابراین بعد از مطالعه این کتاب انتظار می‌رود که اصطلاحات را به طور صحیح بیاموزید. به عنوان مثال: نباید بگوییم رابط‌کاربری محصول من تحت فناوری کیوت سریع طراحی شده است بلکه باید بگویید محصول من تحت فناوری کیوت کوئیک طراحی شده است.

## معرفی کیوت (Qt) ویرایش ۵.۱۳

بدون شک کتابخانه کیوت ویرایش ۵.۱۳ یکی از بهترین توزیع‌های ممکن این کتابخانه برای C++ مدرن در زمان تألیف این کتاب خواهد بود، امید بر آن است که از ابتدای این کتابخانه و در نسخه‌های بعدی آن یعنی نسخه ۶ در کیوت، پشتیبانی چشمگیری از سی‌پلاس‌پلاس ۲۰ که عصر جدیدی را رقم خواهد زد در اختیار علاقه مندان این زبان قرار دهد. در کیوت ۵.۸ بعضی از پایه و اساس کتابخانه تغییر و به کلی عوض شده است برای مثال وابستگی به رابط‌های برنامه‌نویسی مانند اوپن جی ال (OpenGL) با توسعه بسیار بیشتری مواجه بوده که این امکان را فراهم ساخته است تا ۵.۸ ابتدای شروع عصر جدیدی از پشتیبانی نسل‌های جدید رابط‌های برنامه‌نویسی مانند وُلکان (Vulkan) و دایرکت ایکس 12 (Direct3D) در پیش رو

داشته باشد تا بتواند شما را هرچه بیشتر از قبل شیفته خود سازد. البته لازم به ذکر است در نسخه‌های ۵.۱۴ سیستم تولید تصویر بر پایه موتورهای بومی هر یک از پلتفرم‌ها مورد استفاده می‌گیرد.

زمانی که فناوری **کیوت کوئیک** ویرایش دوم (**Qt Quick 2.x**) در نسخه پنجم کیوت در دسترس قرار گرفت، همراه با محدودیت‌هایی که OpenGL 2.0 (ES) و بالاتر از آن را پشتیبانی می‌کند، عرضه شد. فرض بر این بود که با پیش بردن OpenGL، این رابط مسیر حرکت خود را برای تبدیل شدن به API (رابط برنامه‌نویسی نرم‌افزار) شتاب دهنده‌ی سخت افزار، در هر دو زمینه رایانه‌های رومیزی، همراه (موبایل) و سیستم درونی سازی شده (Embedded)، ادامه خواهد داد. با شتابی روز افزون در دو سال اخیر تا به امروز، داستان شتاب گرافیک بسیار پیچیده شده است. یکی از تصوراتی که ما داشتیم این بود که هزینه سخت افزارهای درونی سازی شده (امبدها)، با واحدهای پردازنده‌ی گرافیکی OpenGL، کاهش پیدا خواهد کرد و در آینده در همه جا در دسترس خواهد بود. این فرضیه درست است، اما همزمان دستگاه‌های درونی سازی شده (Embedded) بدون واحدهای پردازنده گرافیکی OpenGL، نیز در دسترس هستند، جایی که مشتریان همچنان آرزو دارند در آن نرم‌افزارهای Quick گسترش یابند. برای حل این مشکل Qt Quick 2D Renderer (سازنده‌ی تصاویر دوبعدی) به عنوان یک پلاگین مجزا برای Qt Quick در ۴/۵ منتشر شد.

همزمان مشخص شد که برنامه‌های Qt Quick بر روی طیف گسترده‌ای از ماشین‌ها، از جمله سیستم‌های قدیمی که اغلب با OpenGL به دلیل از دست دادن یا عدم حضور راه انداز (بویژه در سیستم‌عامل ویندوز) مشکلاتی به همراه دارند، گسترش یافته‌اند. در رابطه با ۴/۵ شرایط، با داشتن قابلیت انتخاب پویا (Dynamic ANGLE) یا یک نرم‌افزار رسم مناسب OpenGL، بهبود یافته است.

به هر حال، این تمامی مشکلات را حل نمی‌کند و نرم‌افزارهای رسم کامل به عنوان یک انتخاب برای سخت افزارهای ارزان قیمت است، مخصوصاً در سیستم‌های درونی سازی شده (Embedded) ها تمامی این‌ها ما را با یک سوال رو به رو می‌کند که چرا تمرکز بر روی پلتفرم‌های بومی، که پتانسیل بهتری در پشتیبانی API ها (از جمله Direct3D) دارند، صورت نمی‌گیرد و چرا سازنده‌های دوبعدی تصاویر به جای نگه داشتن آن در یک ماژول مجزا با فرآیند نصب محربانه، آن را بهبود بخشیده و با مابقی Qt Quick به یکجا جمع نمی‌کنند. بنابراین تصمیم بر این شد که کتابخانه در هسته اصلی خود از رابطه‌ای نسل جدید پشتیبانی و تحت توضیحات تقریباً اشاره شده است کیوت به عنوان یک کتابخانه قدرتمند ویژگی‌های سی‌پلاس‌پلاس پشتیبانی کند.

همانطور که در توضیحات تقریباً اشاره شده است کیوت به عنوان یک کتابخانه قدرتمند ویژگی‌هایی فراتر از ملزمات را در خود جای داده است بنابراین کیوت فراتر از یک کتابخانه همراه با افزونه‌ها و ماژول‌های خاصی در خود به کمک شما می‌آید برای مثال کلاس‌ها و توابع پیشفرض همراه با کیوت برای تمامی سیستم‌عامل‌ها منتشر می‌شود ولی حال اگر قابلیت و هدف خاصی نیاز باشد آن به صورت ماژول بر روی این کتابخانه اضافه می‌شود.

در کل دو نوع ماژول بندی در این کتابخانه وجود دارد که به صورت "ماژول‌های ضروری" و "ماژول‌های اضافی و انحصاری" تعریف می‌شوند که در جداول زیر به آنها اشاره می‌کنیم.

ماژول	توضیحات
Qt Core	کلاس هسته که توسط دیگر ماژول‌ها مورد استفاده قرار می‌گیرد.
Qt GUI	کلاس پایه برای رابط‌کاربری که شامل OpenGL است.
Qt Multimedia	کلاس‌های صوتی و تصویری، رادیو و قابلیت‌های دوربین.
Qt Multimedia Widgets	کلاس‌های مبتنی بر ویجت برای پیاده سازی قابلیت‌های چند رسانه‌ای.

کلاسی برای ایجاد برنامه‌نویسی شبکه‌ی راحت تر و قابل حملتر است.	<b>Qt Network</b>
کلاس‌هایی که شامل JavaScript و QML هستند.	<b>Qt QML</b>
چهارچوب اعلانی برای ساخت برنامه‌های کاربردی بسیار پویا با رابطهای کاربری سفارشی و مدرن.	<b>Qt Quick</b>
قابل استفاده مجدد در Qt به صورت سریع و ایجاد UI بر اساس کنترل برای ایجاد رابط کاربر دسکتاپ به سبک کلاسیک.	<b>Qt Quick Controls</b>
انواع دیالوگ‌ها برای استفاده در برنامه‌های کاربردی.	<b>Qt Quick Dialogs</b>
طرح بندی آیتم‌های مورد استفاده به بر اساس Qt Quick 2.0	<b>Qt Quick Layouts</b>
کلاسی برای یکپارچه سازی پایگاه داده با استفاده از SQL.	<b>Qt SQL</b>
کلاسی برای آزمایش واحد برنامه‌های کاربردی در کتابخانه Qt.	<b>Qt Test</b>
کلاس‌هایی برای توسعه رابط کاربری توسط ویجت تحت کدهای C++	<b>Qt Widgets</b>

به عنوان مثال برای کار با کدهای بسیاری می‌توان نوشت، اما با وجود کیوت برای راحتی کار مازولی با نام Qt Bluetooth را فراهم شده است تا به راحتی برای ارتباط با سخت افزار و بولتوث بپردازید.

برخی از مازول‌های موجود در کتابخانه به صورت زیر مشخص شده است:

توضیحات	سکوی مورد استفاده	سکوی	مازول
کلاس برای برنامه‌های کاربردی است که استفاده از اکتیو ایکس و COM را فراهم می‌کند.		همه	<b>Active Qt</b>
یک راه حل مدیریت به عنوان یک سرویس برای سهولت توسعه بخش مدیریت برای برنامه‌های کاربردی متصل شده و مبتنی بر داده.	همه	همه	<b>(منسوخ شده) Enginio</b>
قابلیتی برای تولید و شبیه سازی سیستم با پشتیبانی تولید تصویر در محیط ۲ بعدی و ۳ بعدی.	همه	همه	<b>Qt 3D</b>
رابطهای برنامه کاربردی پلت فرم خاص برای آندروید را فراهم می‌کند.	Android	همه	<b>Qt Android Extras</b>
دسترسی به سخت افزار بلوتوث را فراهم می‌کند.	Android,iOS, Linux macOS و	همه	<b>Qt Bluetooth</b>
کلاس برای نوشتن برنامه‌های چند رشته‌ای بدون نیاز به برنامه‌نویسی از سطح پایین		همه	<b>Qt Concurrent</b>
فعال کننده OpenGL مانند رسام ۳ بعدی و استفاده از آن در اپلیکیشن‌هایی که بر پایه فناوری Qt Quick و JavaScript تولید می‌شوند.	همه	همه	<b>Qt Canvas 3D</b>
کلاس‌های برای برقراری ارتباط بین فرایند از طریق پروتکل‌های D-BUS.		همه	<b>Qt D-Bus</b>
تأثیرات گرافیکی برای استفاده در کیوت کوئیک ۲/۰		همه	<b>Qt Graphical Effects</b>
پلاگین برای فرمتهای تصویری اضافی WBMP, TGA, MNG :: TIFF		همه	<b>Qt Image Formats</b>

رابطهای برنامه کاربردی پلت فرم خاص برای سیستم عامل macOS را فراهم می‌کند.	macOS	همه	<b>Qt Mac Extras</b>
دسترسی به ارتباطات نزدیک میدان سخت افزارهای مجهر به (NFC) را فراهم می‌کند.	Android و Linux	همه	<b>Qt NFC</b>
کلاس پشتیبانی از موتور OpenGL توجه : برنامه‌های که توسط نسخه ۴ نوشتۀ شده‌اند پشتیبانی می‌شوند بنابراین برای استفاده در کدهای جدید از کلاس QOpenGL در <a href="#">Qt GUI</a> استفاده کنید.		همه	<b>(منسوخ شده) Qt OpenGL</b>
کلاسی برای کیسوله سازی اطلاعات خاص را در انواع سیستم عامل‌ها فراهم می‌کند.		همه	<b>Qt Platform Headers</b>
دسترسی به کلاس‌های موقعیت، ماهواره‌ای و نظارت بر منطقه را فراهم می‌کند.	Android,iOS, OS X,Linux,WinRT.	همه	<b>Qt Positioning</b>
کلاسی که امکان پرینت را به صورت قابل حمل تر فراهم می‌کند.		همه	<b>Qt Print Support</b>
فعال کننده سیستم خرید در محصولات کیوت.	Android, iOS, و OS X.	همه	<b>Qt Purchasing</b>
کلاس Qt برای نسخه ۴ فراهم شده است که برای استفاده در Qt QUICK است و در نسخه‌های جدید نیز پشتیبانی می‌شود.		همه	<b>Qt Declarative</b>
فراهم کننده سبک ایجاد کننده برنامه از نوع QML جهت تولید و سازگاری کامل با انواع دستگاه‌های موبایل و غیره...		همه	<b>Qt Quick Controls 2</b>
فراهم کننده مجموعه‌ای از کنترل‌های اختصاصی در طراحی با رابط .Qt Quick		همه	<b>Qt Quick Extras</b>
فراهم کننده کلاس‌های widget در سی‌پلاس‌پلاس برای رابط کاربری تحت Qt Quick			<b>Qt Quick Widgets</b>
کلاسی برای فراهم ساختن برنامه‌های کاربردی اسکریپتی است که در نسخه ۴ نیز قابل پشتیبانی است بنابراین برای استفاده از کلاس QJS استفاده نمایید.		همه	<b>(منسوخ شده) Qt Script</b>
اجزای اضافی برای برنامه‌های کاربردی که با استفاده با ماژول Qt Script نوشته می‌شوند.		همه	<b>(منسوخ شده) Qt Script Tools</b>
دسترسی به سخت افزار سنسور و تشخیص اشاره و حرکت را فراهم می‌کند.	Android,Qt for iOS,WinRT	همه	<b>Qt Sensors</b>
دسترسی به سخت افزار و پورت سریال مجازی را فراهم می‌کند.	Windows,Linux, و OS X	همه	<b>Qt Serial Port</b>
کلاس برای نمایش محتویات فایل‌های SVG. پشتیبانی از زیر مجموعه‌ای از SVG 1.2 استاندارد ریز است.		همه	<b>Qt SVG</b>
دسترسی به آبجکت‌های QObject یا HTML را از طرف HTML توسط کلاینت JavaScript/JAVASCRIPT را فراهم می‌کند.	همه	همه	<b>Qt WebChannel</b>
امکان دسترسی و اجرای API های برنامه‌های کاربردی در موتور مرورگر کرومیوم را فراهم می‌کند.	Windows,Linux, و OS X.	همه	<b>Qt WebEngine</b>
ارتباطات WebSocket سازگار با RFC 6455 را فراهم می‌کند.	همه	همه	<b>Qt WebSockets</b>
نمایش دهنده محتوای وب در یک برنامه QML با استفاده از رابطهای برنامه‌نویسی بومی در پلتفرم بدون نیاز به استفاده کامل از سیستم پشتیبانی از سکویی که دارای یک وب انجین بومی است.		همه	<b>Qt WebView</b>

امکان ارتباط با API های خاص پلتفرم ویندوز را فراهم می کند.	Windows	همه	<b>Qt Windows Extras</b>
امکان ارتباط با API های خاص پلتفرم X11 را فراهم می کند.	Linux/X11	همه	<b>Qt X11 Extras</b>
پیاده سازی C ++ از SAX و DOM را که مختص XML است را فراهم می کند.			<b>Qt XML</b>
پشتیبانی از XPath، XQuery، XSLT و اعتبار سنجی مدل XML.	همه	همه	<b>Qt XML Patterns (منسخه شده)</b>
ماژول های زیر تحت مجوز GPLv3 افزوده شده اند.	سکوی مورد استفاده	سکوی توسعه	<b>ماژول</b>
ارائه دهنده کامپوننت های رابط کاربری جهت نمایش نمودارهای بصری جذاب در انواع داده های استاتیک و پویا.		همه	<b>Qt Charts</b>
کامپوننت ارائه دهنده رابط کاربری جهت نمایش دادن داده های ۳ بعدی خیره کننده.	همه	همه	<b>Qt Data Visualization</b>
چهار چوبی برای اجرای روش های مختلفی از ورودی ها و همچنین یک صفحه کلید مجازی تحت QML با قابلیت سفارشی ساطع کلیدها و سبک های آن.	Linux و Windows Boot to Qt های و پروژه	همه	<b>Qt Virtual Keyboard</b>
امکان استفاده از رابط های تحت فناوری Qt Quick را در دستگاه هایی میسر می سازد که از OpenGL پشتیبانی نمی کنند.		همه	<b>Qt Quick 2D Renderer</b>
ماژول های زیر تحت مجوز های تجاری و LGPL افزوده شده اند.	سکوی مورد استفاده	سکوی توسعه	<b>ماژول</b>
ابزاری برای سرعت بخشیدن و راحتی و یکپارچه سازی کامل نرم افزار در دستگاه های تعبیه شده (Embedded) است که دارای ویژگی های دیگری نیز است.	همه	همه	<b>Qt for Device Creation</b>
امکان کامپایل فایل های qml. را در داخل سورس باینتری برنامه فراهم می سازد تا روشی باشد برای بهبود سرعت و امنیت کدهای آن.	همه	همه	<b>Qt Quick Compiler</b>
ماژول های به عنوان پیش نمایشی از یک فناوری افزوده شده اند.	سکوی مورد استفاده	سکوی توسعه	<b>ماژول</b>
فعال سازی و ارائه قابلیت پشتیبانی از دسته بازی را به برنامه های تحت فراهم می کند.	همه	همه	<b>Qt Gamepad</b>
دسترسی به Serial Bus را فراهم می کند. توجه داشته باشید که در حال حاضر این ماژول از Mod bus و Can bus پشتیبانی می کند.	همه	همه	<b>Qt Quick Qt Serial Bus</b>
دسترسی به ویژگی های متن به گفتار را فراهم و پشتیبانی می کند.	همه	همه	<b>Qt Speech</b>
چهار چوبی را جهت توسعه Wayland فراهم و ارائه می کند.	همه	همه	<b>Qt Wayland Compositor</b>
فراهم کننده وضعیت ماشین در فایل های SCXML است.	همه	همه	<b>Qt SCXML</b>
امکان تولید خروجی در مرورگر بر اساس فناوری wasm.	همه	همه	<b>Qt WebAssembly</b>

در صورتی که شما از ابزار qmake جهت ساخت پروژه استفاده می‌کنید کافی است کد زیر را در فایل pro. پردازه قرار دهید، در حالت به صورت خودکار مازولهای Qt GUI و Qt Core به صورت پیشفرض به پردازه شما افزوده خواهند شد.

QT +=gui

## آشنایی با محیط توسعه، نصب و راهاندازی همراه با پیکربندی کیت (Kit) در آن

یکی از بزرگترین مشکلات که در بین جامعه برنامه‌نویسان سی‌پلاس‌پلاس و مخصوصاً کتابخانه Qt رُخ می‌دهد، مواجه شدن با خطاهای و مشکلاتی است که در مرحله آماده‌سازی و دریافت آن است. بنابراین جهت پیکربندی و نصب و راهاندازی مناسب‌ترین نسخه در این کتاب اطلاعاتی را فراهم کرده‌ایم که بتوانید بر اساس نیاز در کمترین زمان ممکن و بدون مراجعه به هیچ مرجعی تشخیص دهید که کدام نوع پیکربندی از کیوت را باید راهاندازی نمایید.

من نویسنده این کتاب (کامبیز اسدزاده) به شما اطمینان می‌دهم در صورتی که توضیحات مربوطه را به درستی همراه با صبر و حوصله پیش ببرید نیازی برای سر درگم شدن و مراجعه به انجمان‌ها و مستندات خارج از این کتاب نخواهید داشت.

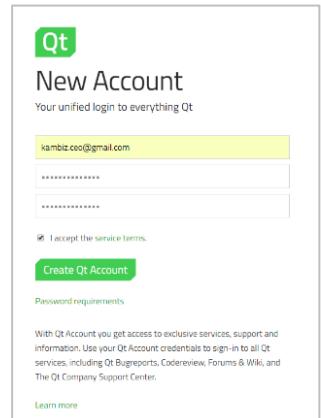
خوبشخانه این کتابخانه در تمامی سکوهای موجود قابل استفاده است که به صورت زیر معرفی گردیده است.

- (Unix / Linux) برای خانواده Qt/X11
- iOS برای مکینتاش قابلیت پشتیبانی در Mac OS و
- Windows CE برای ویندوز Qt/Windows
- Qt/Embedded وسائل همراه (PDA, تلفن هوشمند و غیره)
- Windows CE برای WinCE Qt/WinCE
- Java برای Qt Jambi
- Qt Extended برای سیستم‌عامل لینوکس نسخه وسائل همراه

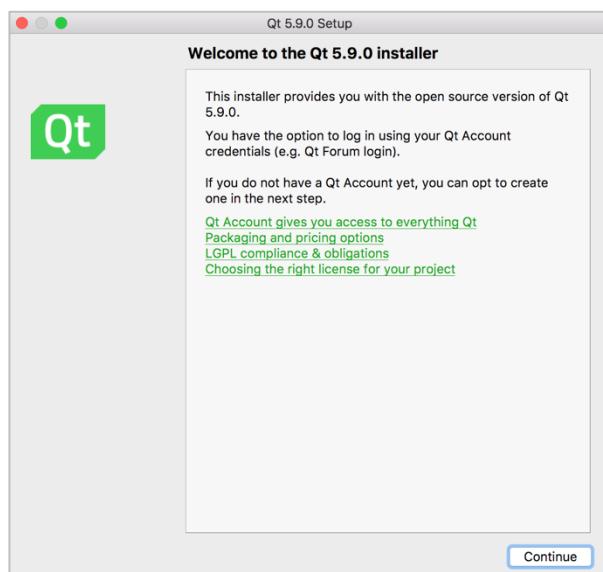
## نصب و راهاندازی محیط Qt

همانطور که می‌دانید معمولاً برای استفاده کتابخانه‌های C++ نیاز است تا آن‌ها را کامپایل و بعد در محیط توسعه خود وارد و سپس از آن‌ها استفاده نماییم، این روش در این کتابخانه نیز صدق می‌کند با این تفاوت که توسعه‌دهنده رسمی آن نسخه‌های کامپایل شده آن را در اختیار کاربران قرار می‌دهند، بنابراین نیازی نیست این کتابخانه را کامپایل نماییم مگر اینکه در موارد خاص ترجیح دهیم با تنظیمات خاصی از آن خروجی بگیریم که در این صورت از سایت رسمی آن می‌توانیم منبع کُد آخرین نسخه و حتی نسخه‌های پیشین آن را دریافت کنیم.

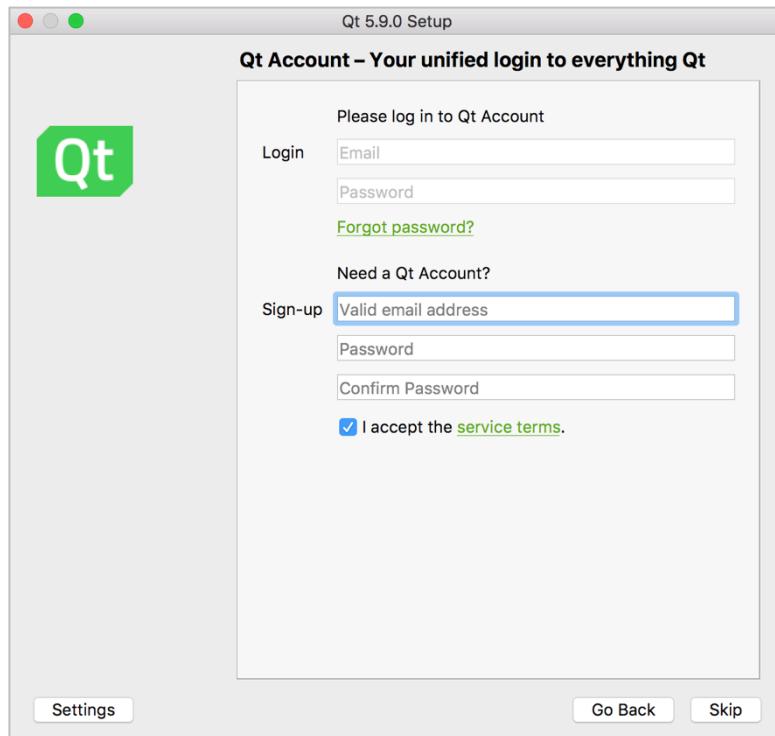
این کتابخانه در یک بسته کامل همراه با تمامی مازولهای همچنین محیط Qt Creator فراهم شده است، بنابراین بعد از دریافت نسخه مورد نظر برای نصب آن از نسخه ۵.۵ به بعد می‌توانید یک حساب کاربری در وبسایت qt.io داشته باشید، بنابراین با ایجاد یک حساب کاربری در این وبسایت به آدرس <https://login.qt.io/register> ایجاد و یا هنگام نصب برنامه در مراحل ابتدایی فرم ثبت نام را می‌توانید پُر کنید.



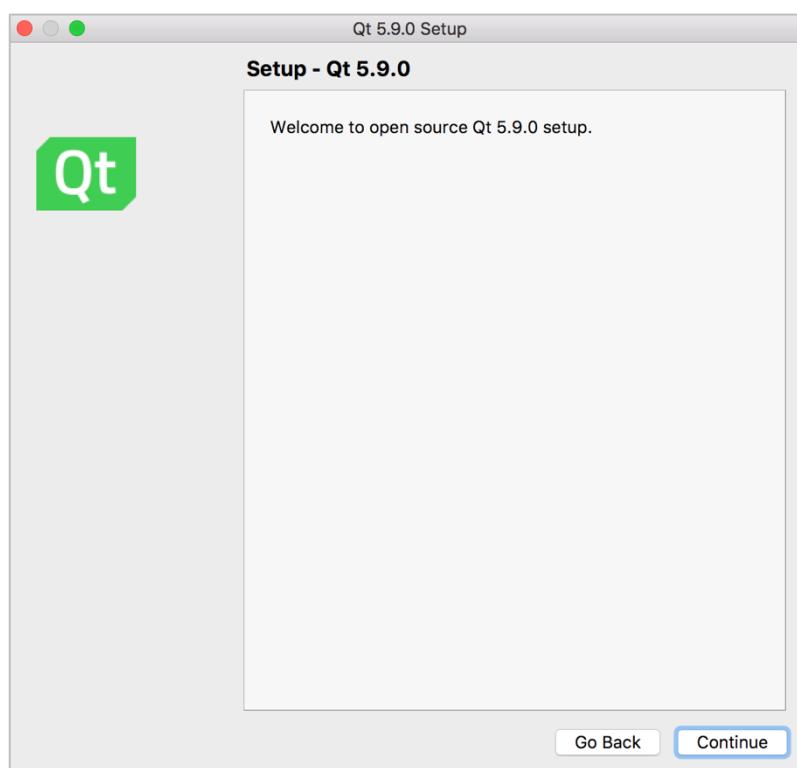
حال بر فرض اینکه ما حساب کاربری خود را ساخته‌ایم، حساب کاربری بعدها در بعضی از پیکربندی‌های آنلاین نیاز خواهد شد کافی است فایل اجرایی نصبی آن را اجرا نماییم تا با محیط زیر روبرو شویم:



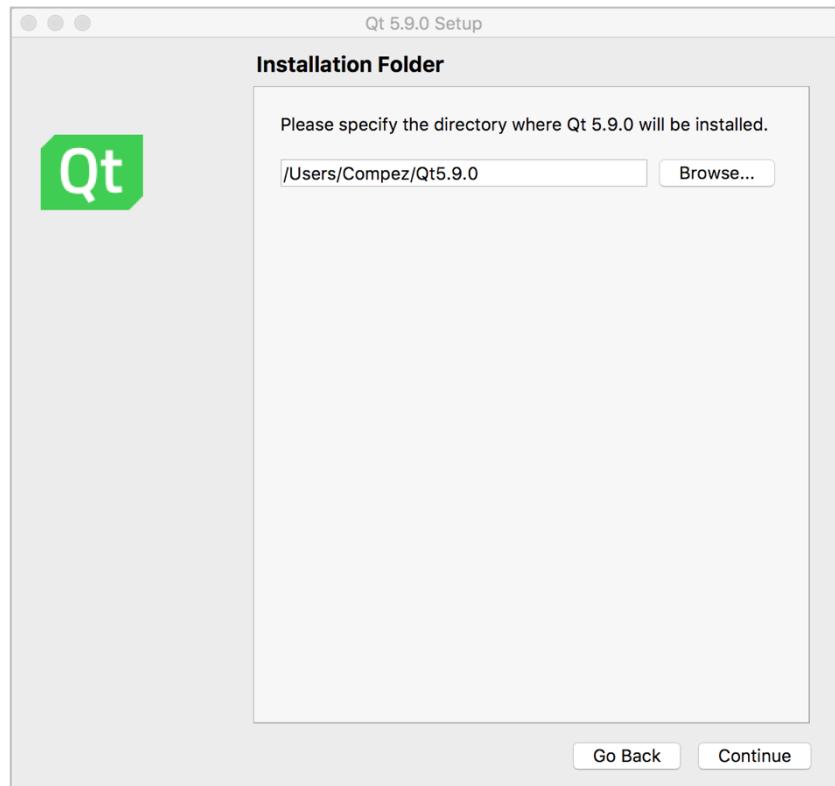
بعد از این مرحله گزینه Next را زده و وارد مرحله بعدی خواهیم شد که در این بخش باید نام کاربری و رمز عبوری را که ساخته‌ایم وارد نماییم، در صورتی که نمی‌خواهید با حساب کاربری وارد مرحله نصب شوید گزینه skip را انتخاب و این مرحله را رد کنید.



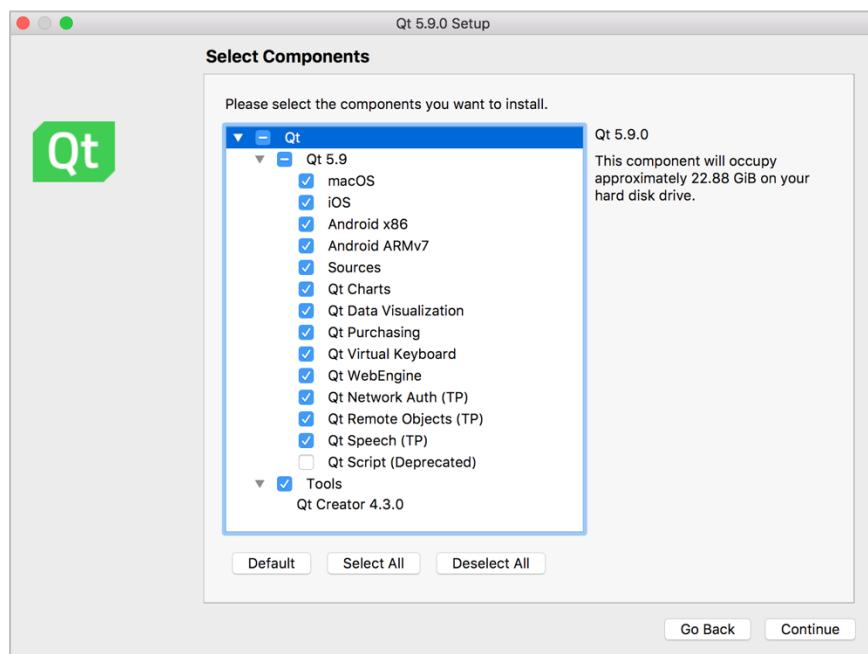
در صورتی که اطلاعات حساب کاربری را وارد کنید اطلاعات شما بر روی سرور شرکت کیوت ارسال خواهد شد، در غیر این صورت بر روی گزینه Skip کلیک کنید تا مرحله بعدی جهت یک خوش آمدگویی برای ادامه نصب نمایان شود:



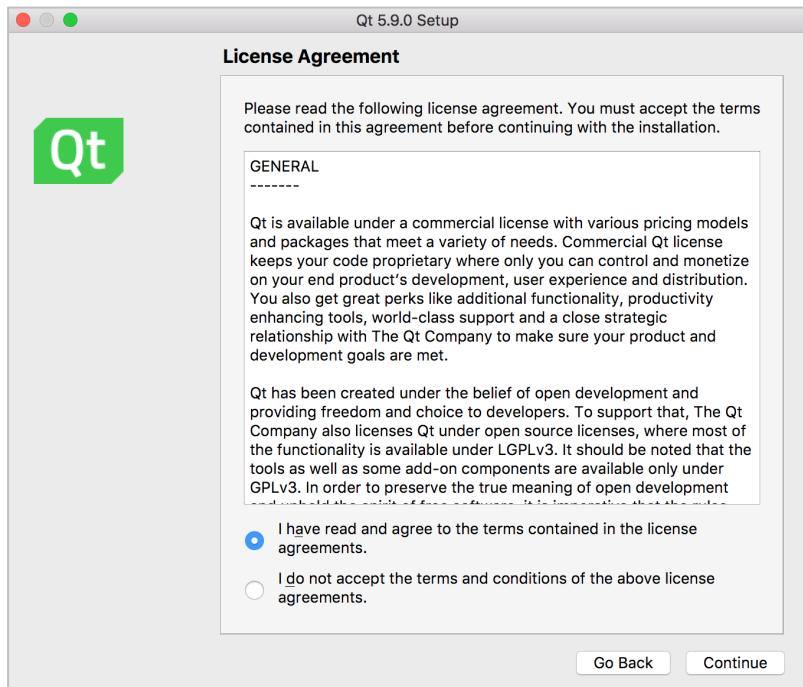
سپس در مرحله بعدی باید مسیری که قرار است کتابخانه و محیط توسعه در آن نصب شود را مشخص کنید:



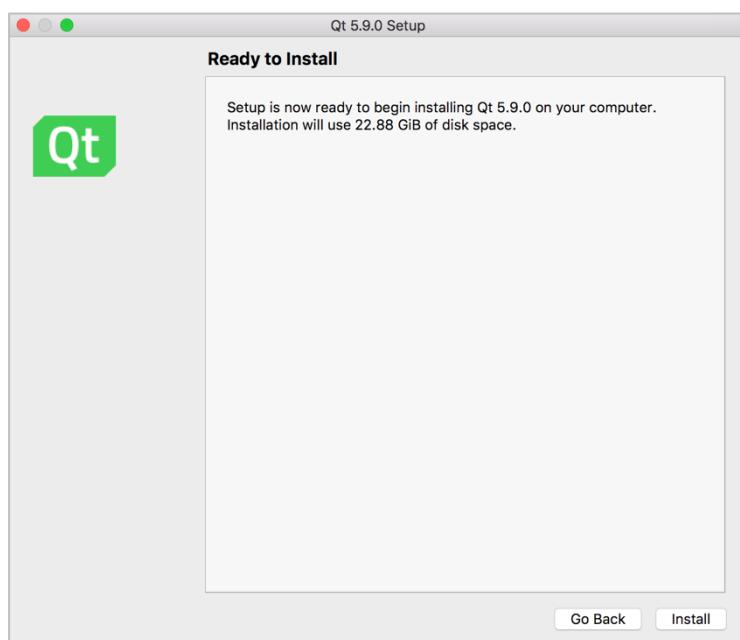
در مرحله بعدی تمامی موارد قابل انتخاب را انتخاب کرده و ادامه دهید:



در ادامه با پذیرش مجوزهای مربوط به این کتابخانه طبق تصویر بعدی خواهیم توانست با ادامه و در نهایت نصب آن را به پایان برسانیم.



توجه داشته باشید که حتماً فضای کافی جهت نصب و پیاده سازی این کتابخانه را داشته باشید که حداقل 4.37 گیگابایت برای نصب این کتابخانه مورد نیاز خواهد بود، سپس بعد از نصب محیط برنامه‌نویسی همراه با تمامی مازول‌ها آماده است.



**نکته:** توجه داشته باشید که برای استفاده از نسخه‌های مختلف کیوت نباید آن‌ها را مجدداً در مسیری که قبلاً نسخه‌ای از آن را داشته‌اید نصب کنید، هرچند در زمان نصب اجازه نصب در پوشه‌ای که از قبل وجود داشته است را نخواهد داد. لذا پیشنهاد، سفارشی سازی مسیر نصب است.

کتابخانه Qt از پلتفرم‌های مختلف با کامپایلرهایی که شامل نسخه‌های متفاوتی هستند پشتیبانی می‌کند، بنابراین بسیاری از کاربران برای اینکه انتخاب صحیح را در رابطه با محیط توسعه مرتبط با کیوت انجام دهند پیشنهاد می‌شود فقط طبق توضیحات عمل کنند. قبل از هر چیز به چند نکته بسیار مهم اشاره می‌کنیم که می‌بایست قبل از اقدام به دریافت کیوت به آن‌ها توجه کنید.

۱. از نوع و توزیع سیستم‌عاملی که شما تصمیم دارید در محیط آن به کد نویسی بپردازید را مطمئن شوید.

۲. به ویرایش کامپایلر موجود بر روی سیستم‌عامل توجه کنید.

- برای مثال در صورتی که بر روی سیستم‌عامل ویندوز هستید باید به این توجه داشته باشید که محیط Visual Studio شامل چه نسخه‌ای است.

- معمولًاً ویرایش ۲۰۱۳ شامل کامپایلر نسخه ۱۲ بوده و نسخه‌های ۲۰۱۵ و ۲۰۱۶ شامل کامپایلر نسخه ۱۴ است که به ترتیب نسخه از نسخه سی‌پلاس‌پلاس پشتیبانی می‌کند. جدیداً نسخه ۲۰۱۷ محیط توسعه آن مجهز به استاندارد ۱۷ و نسخه ۲۰۱۹ شامل ویرایش ۱۹ کامپایلر مایکروسافت سی++ است.

- در صورتی که در سیستم‌عامل مک‌ینتاش تصمیم به کدنویسی دارید باید آن را مجهز به آخرین نسخه Xcode نمایید در غیر اینصورت قادر به کامپایلر برنامه نخواهید بود.

- در سیستم‌عامل لینوس از نصب کامپایلر GCC بر روی آن مطمئن شوید.
- ۳. هدف خود را برای توسعه و برنامه‌نویسی مشخص کنید، برای مثال هدف من برنامه‌نویسی و توسعه نرم‌افزار در محیط دسکتاپ است که بر روی ویندوز قرار است اجرا شود. اگر شما هدف دیگری دارید مثلاً قرار است برنامه‌ای تولید کنید که در پلتفرم موبایل تحت سیستم‌عامل اندروید اجرا شود، می‌بایست بسیار دقیق کنید و به تمامی پیش‌نیازاتی که لازم است توجه نمایید.

برای این منظور روش‌های دریافت و نصب نسخه مناسب به ترتیب زیر خواهد بود.

## پیکربندی کیت‌ها در macOS

### روش نصب و راهاندازی محیط توسعه در سیستم‌عامل مک:

- ابتدا مطمئن شوید که سیستم‌عامل نصب شده بر روی دستگاه مک شما مجهز به آخرین نسخه موجود باشد.
- محیط توسعه Xcode را از لینک <https://developer.apple.com/xcode/> دریافت کنید. دقیق کنید که حتماً باید آخرین نسخه ممکن را دریافت و نصب کنید، در غیر اینصورت مجوز توسعه و ساخت برنامه را بر روی دستگاه‌های مک بوک، آیفون و آی پد نخواهید داشت. این یکی از حساسیت‌های شرکت آپل از لحاظ امنیتی است که توصیه می‌کند برای حفظ امنیت محصولات همیشه باید ابزار توسعه به روز باشد.
- بعد از نصب Xcode بر روی مک برای اطمینان کامل دستور `gcc --version` را اجرا کنید. در صورتی که همه چیز به خوبی پیش رفته باشد نتیجه‌های مشابه نتیجه زیر خواهید داشت.

```
Kambizs-MacBook-Pro:~ Compez$ gcc --version
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/include/c++/4.2.1
Apple LLVM version 8.1.0 (clang-802.0.42)
Target: x86_64-apple-darwin16.6.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
```

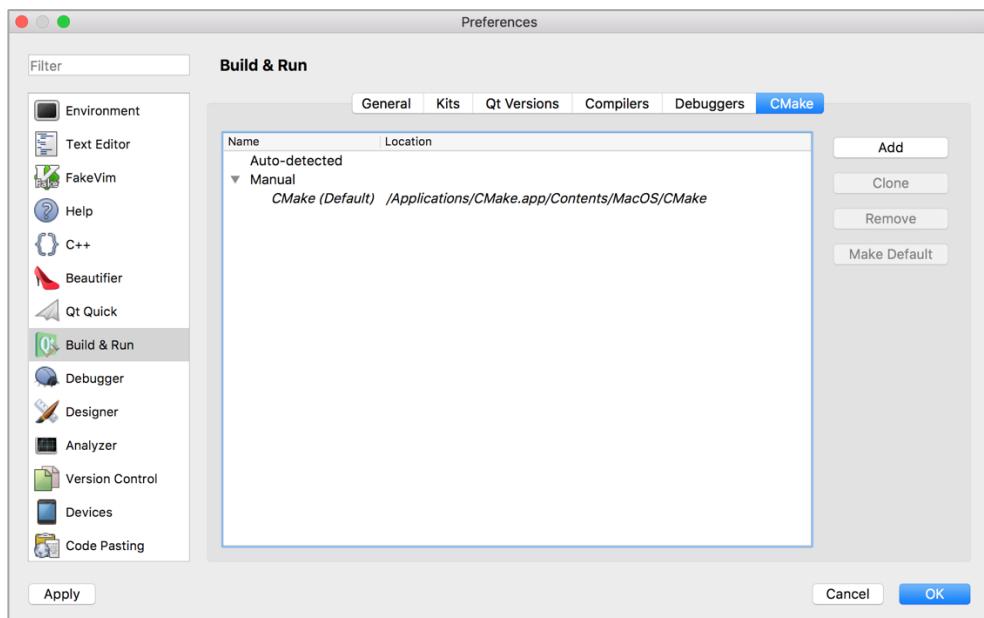
پیغام مربوطه نشان می‌دهد که Xcode به درستی نصب و کامپایلر GCC نیز به خوبی شناسایی شده است. در زمان تالیف این کتاب نسخه محیط توسعه Xcode ویرایش ۸ نوع بتا است. دقیق کنید ممکن است در صورتی که از Xcode نسخه beta استفاده می‌کنید خطایی در هماهنگی Qt

رخ دهد که ناشی از عدم شناسایی مسیر پیشفرض Xcode برای کیوت است در این صورت کافی است دستور زیر را در ترمینال سیستم عامل اجرا کنید.

```
#sudo xcode-select -switch /Applications/Xcode.app/Contents/Developer
```

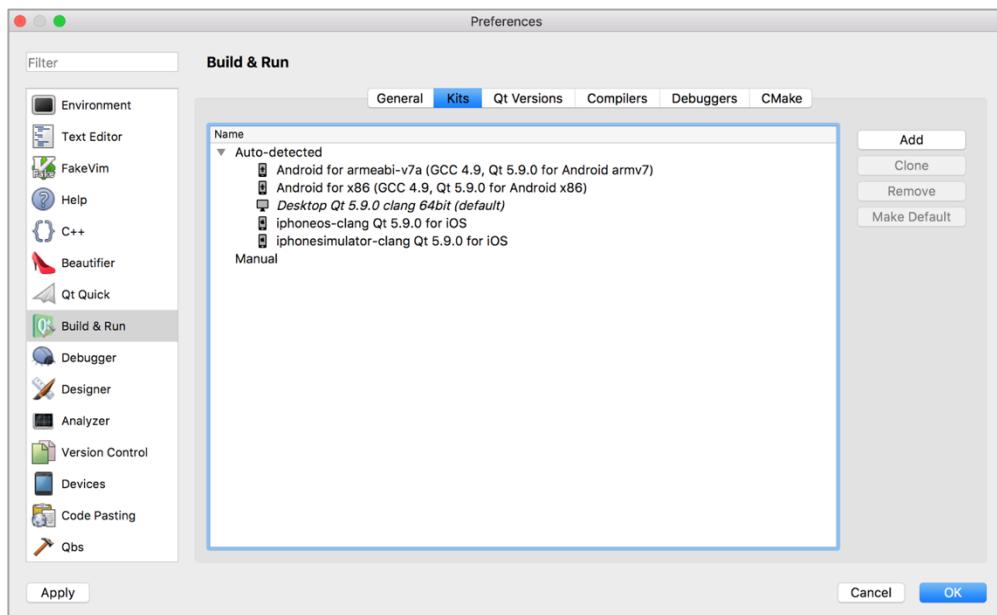
در این صورت بدون خطا قادر به کامپایل و ساخت برنامه تحت کیوت بر روی مک خواهد بود.  
کیت‌ها (بسته‌های) ساخت برنامه تحت کامپایلر و نوع پلفترم هستند که برای مدیریت آن‌ها باید نوع کامپایلر و نوع معماری سیستم را مورد توجه قرار داد. در کیوت این موارد به طور خودکار بعد از نصب محیط Xcode قابل شناسایی است. برای اینکه صحت عملیات نصبی در مک مطمئن شویم محیط Qt Creator را اجرا کرده و به منوی **Build & Run Preferences** وارد شده و زبانه **Qt Creator** را انتخاب خواهیم کرد تا مطابق تصویر زیر وارد زبانه‌های مربوطه شویم.

- زبانه **CMake** جهت تنظیمات و سفارشی سازی ابزار cmake است که به طور پیشفرض Qt از ابزار qmake به جای استفاده می‌کند. در صورتی که بخواهید از ابزار cMake استفاده کنید باید در این بخش پیکربندی مربوط به آن را انجام دهید. اگر مایل باشید برای این کار کافی است وارد سایت رسمی این ابزار شوید : <https://cmake.org/download> آن را دریافت و در محیط توسعه خود فراخوانی کنید.

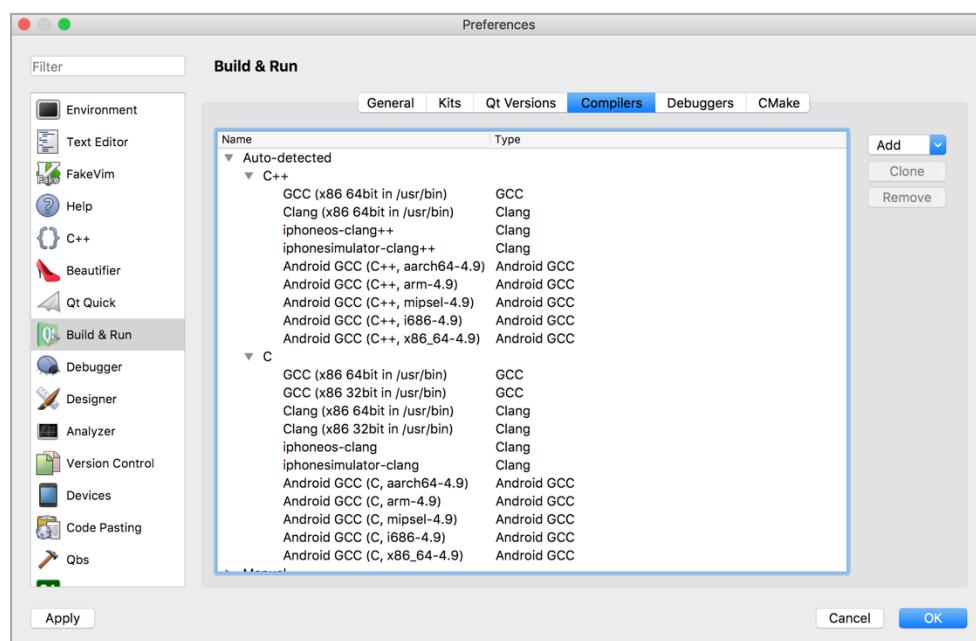


- زبانه **Debuggers** شامل دیباگرهای پیشفرض و سفارشی سازی آن‌ها است.
- زبانه **Compilers** نیز شامل کامپایلرها (همگردان‌های) موجود بر روی سیستم است.
- زبانه **Qt Version** بخشی است که به شما اجازه می‌دهد نسخه‌های مختلفی از کیوت را بر روی محیط توسعه خود داشته باشید که شامل ابزار qmake ای است که متناسب با سیستم مربوطه ثبت و سازگار با نسخه موجود است در صورتی که qmake مربوطه متناسب با کامپایلر موجود بر روی سیستم شما نباشد امکان هماهنگی کیت وجود نخواهد داشت. بنابراین موقع دانلود نسخه مورد نظر حتماً در نظر داشته باشید که بر اساس نسخه Qt مربوطه و نوع کامپایلر موجود بر روی سیستم آن را دریافت کنید.

- در نهایت زبانه **Kit** شامل کیت‌هایی است که بر اساس ابزارهای مرتبط شناسایی و ایجاد شده‌اند.



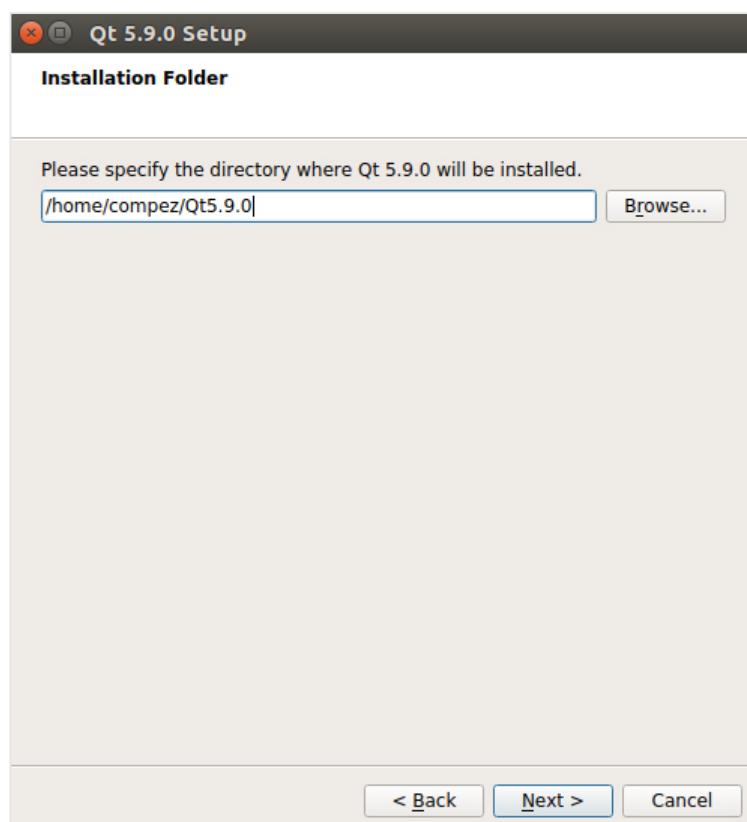
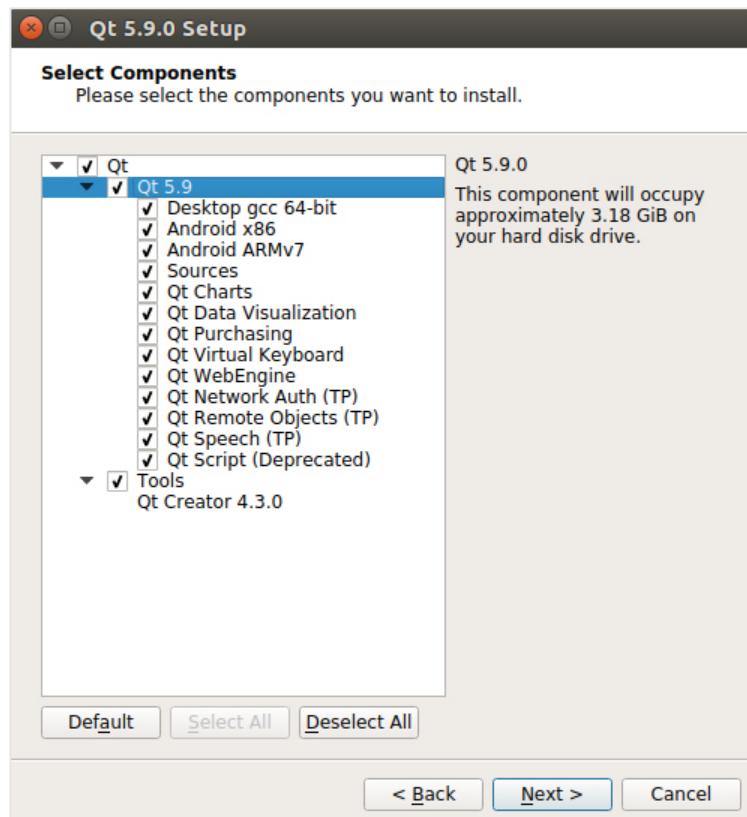
همانطور که مشخص است بسته‌های مربوطه به کامپایلر و پلتفرم‌های موجود بر روی مک شناسایی شده‌اند. در صورتی که Xcode نصب نباشد و به درستی تنظیم نشده باشد در این بخش و بخش کامپایلرها شناسایی نخواهد شد.



در تصویر فوق کامپایلرهای Clang و GCC با معماری ۶۴ و ۳۲ بیت شناسایی شده و قابل استفاده می‌باشند.

## نصب کیوت و مدیریت کیت‌ها در Linux

بر فرض اینکه محیط لینوکس به طور خام باشد ابتدا نیاز است تا کامپایلر GCC را همراه با چند کتابخانه مورد نیاز نصب و بعد از آن محیط Qt را راهاندازی نماییم که همانند مراحل موجود در ویندوز و مک خواهد بود.



برای اینکار کافی است تحت دستور زیر کامپایلر مربوطه را بر روی ایستگاه یونیکس در محیط ترمینال نصب نمایید:

```
sudo apt-get install gcc
```

بعد از نصب و به اتمام رسیدن کامپایلر بر روی لینوکس کدهای زیر را در ترمینال اجرا خواهیم کرد. همچنین می‌توانید گزینه بعدی مربوط به `g++` را اجرا کنید. در صورتی که قبل از این عمل کیوت را نصب کنید با خطای `error: g++: Command not found` مواجه خواهید شد.

```
sudo apt-get install build-essential
```

گزینه بعدی مربوط به کتابخانه پیکربندی فونت‌ها خواهد بود:

```
sudo apt-get install libfontconfig
```

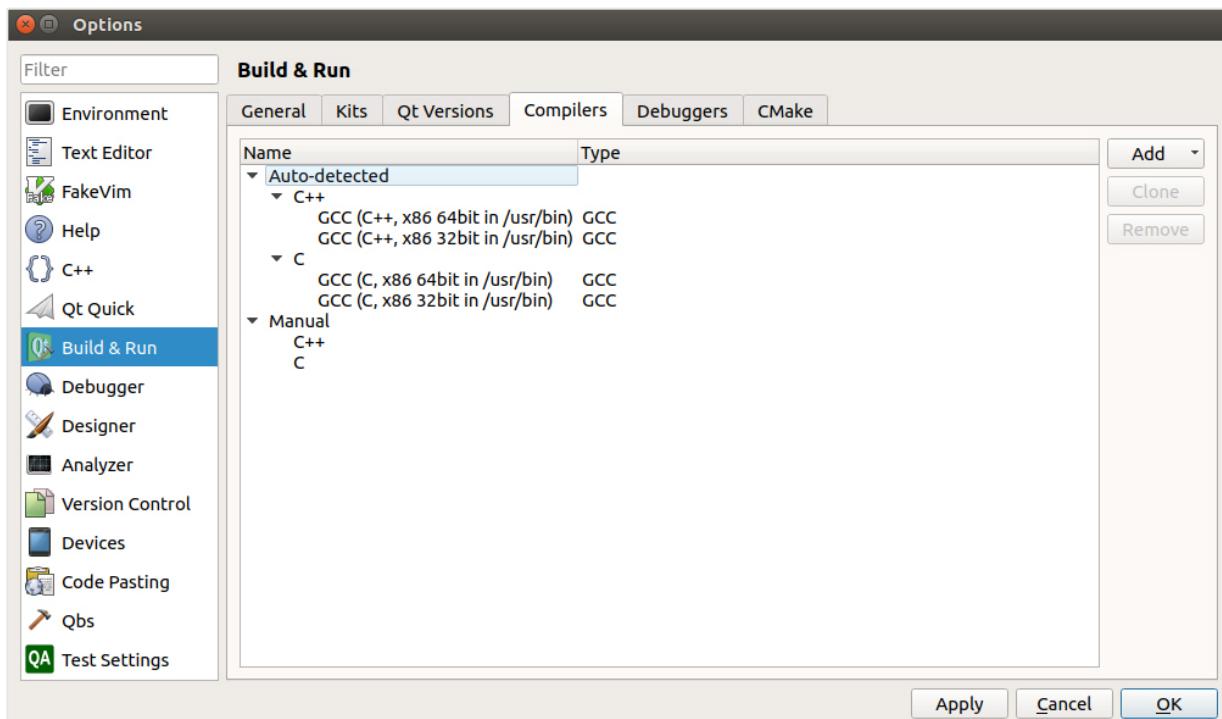
دستور بعدی مربوط به پیش نیازی از کتابخانه OpenGL است و در صورتی که قبل از نصب آن اقدام به نصب کیوت کرده باشید با خطای `error: GL/gl.h: No such file or directory` مواجه خواهید شد.

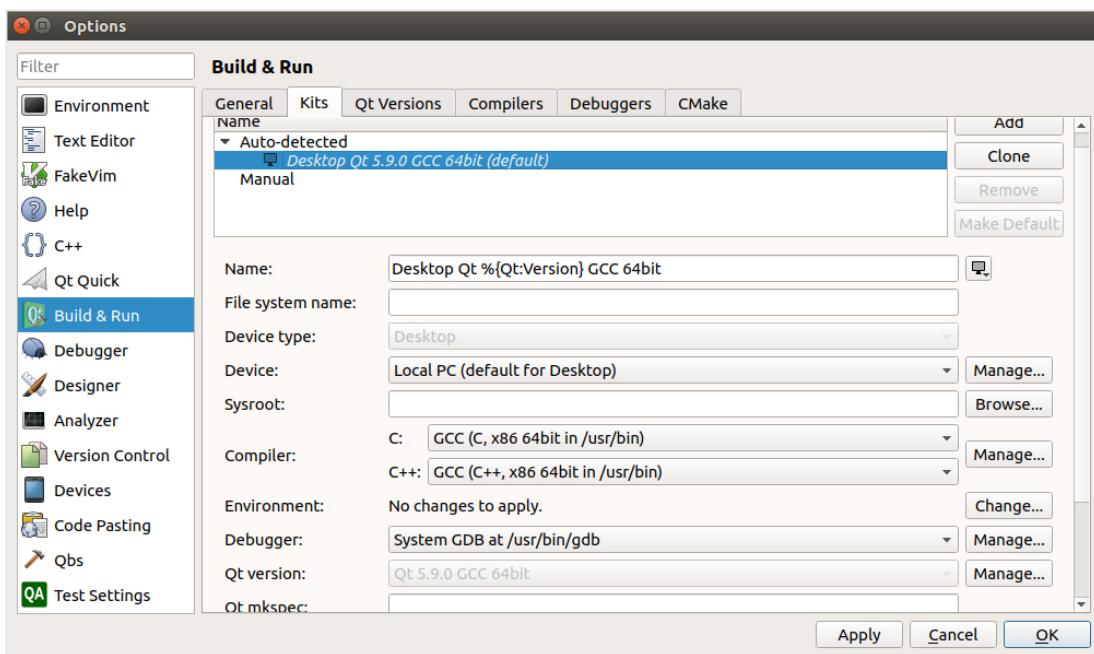
```
sudo apt-get install mesa-common-dev libglu1-mesa-dev
```

در نهایت کتابخانه کیوت را در مسیری که دانلود شده است مجوز داده و سپس نصب می‌کنیم.

```
chmod +x qt-opensource-linux-x64-5.9.0.run  
. ./qt-opensource-linux-x64-5.9.0.run
```

حال محیط توسعه کیوت و کتابخانه آماده است. کافی است وارد بخش کیت شده و همانند توضیحات موجود در مک تنظیمات مورد نظر را در اختیار داشته باشید. وارد زبانه Compilers شوید و لیست کامپایلرهای فعلی را مشاهده کنید.

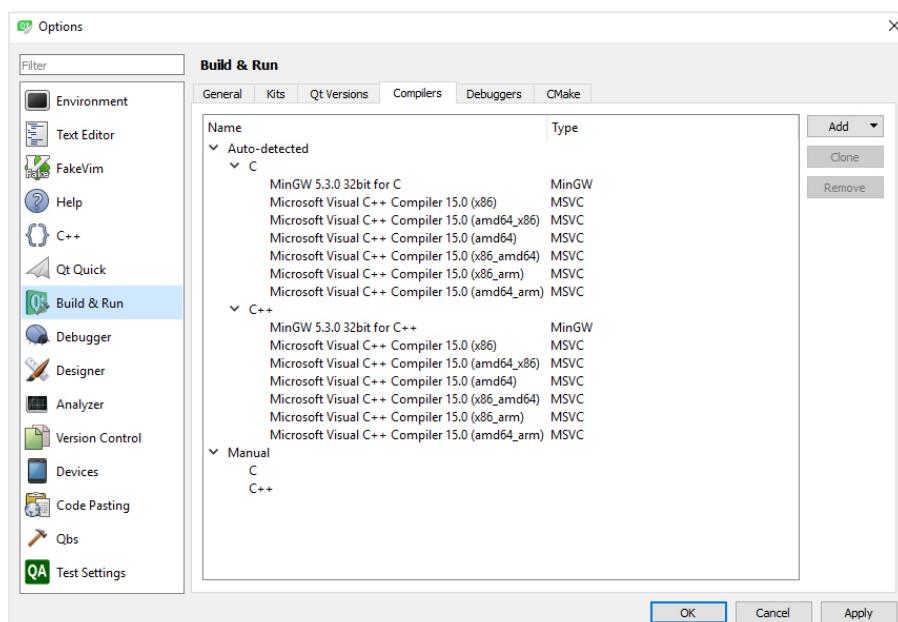




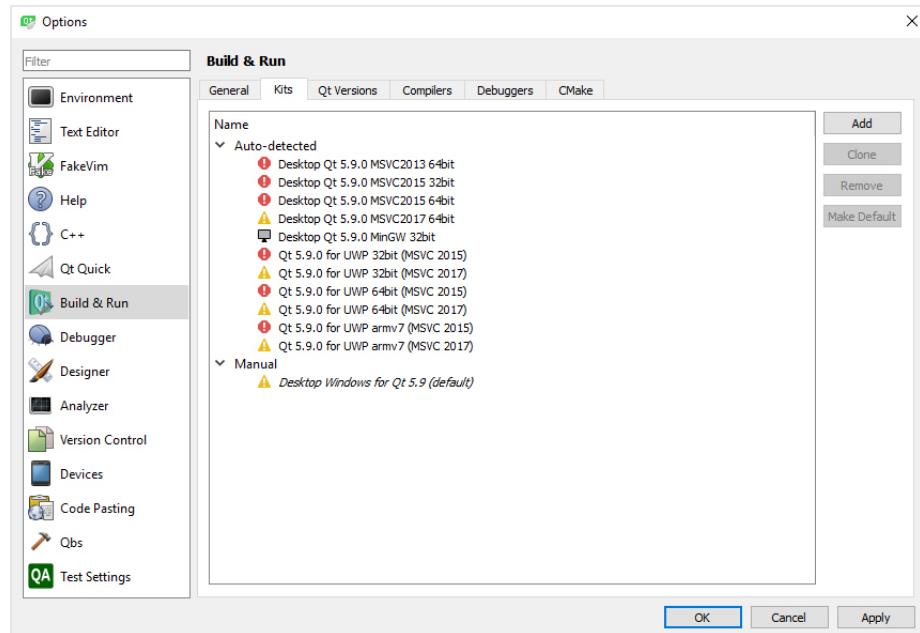
## نصب کیوت و مدیریت کیت‌ها در Windows

- در ویندوز برای پیاده سازی محیط توسعه مورد نظر، نیاز به نصب SDK مخصوص Visual C++ یا کیت Visual Studio است. بنابراین شما باید یکی از نسخه‌های ۲۰۱۳، ۲۰۱۵، ۲۰۱۷ و یا ۲۰۱۹ را بر روی ویندوز نصب کرده باشید. در این کتاب ما از نسخه ۲۰۱۷ استفاده کردیم. بنابراین بعد از پیکربندی، دسترسی به کامپایلرهای نسخه ۲۰۱۷ فراهم خواهد شد.

بعد از نصب محیط Visual Studio 2017 اقدام به نصب کیوت کنید. تمامی مراحل را در نصب ادامه دهید و برای پیکربندی کیت همانند نسخهٔ مک و لینوکس وارد محیط تنظیمات شوید، در ویندوز به جای Tools و Preferences Setting شوید و از زبانه Build & Run گزینه Compilers را انتخاب کنید.



همانطور که مشخص است کامپایلرهایی که شناسایی شده‌اند در لیست موجود هستند. نسخه ۱۵.۰ مختص کامپایلرهای MSVC2017 است که اگر به زبان Kits مراجعه کنید با نتیجهٔ زیر مواجه خواهد شد، که نشان می‌دهد نسخه‌های مربوط به MSVC2017 و MinGW آماده استفاده هستند.



## روش Qt Online Installers

در این حالت شما فایل نصب کننده آنلاین را دریافت و نرم‌افزار نصب کننده به صورت خودکار بر اساس نوع پلتفرم شما یک نسخه مناسب از Qt را بر روی سیستم شما نصب می‌کند که معمولاً در زمان ناپایداری اینترنت این روش پیشنهاد نمی‌شود.

نصب به صورت آنلاین			
اندازه فایل	لينك دریافت	معماری	پلتفرم
۳۱ مگابایت	برای دریافت می‌توانید کلیک کنید	X64 - 64Bit	لينوکس
۳۳ مگابایت	برای دریافت می‌توانید کلیک کنید	X86 - 32Bit	لينوکس
۱۷ مگابایت	برای دریافت می‌توانید کلیک کنید	X86 - 32Bit	ویندوز
۱۲ مگابایت	برای دریافت می‌توانید کلیک کنید	X64 - 64Bit	مکینتاش

## روش دوم Source packages & Other releases

در این شیوه شما کدهای منبع کتابخانه را دریافت و با پیکربندی و کامپایل مجدد آن می‌توانید کتابخانه را بر روی سیستم خود نصب نمایید. معمولاً این شیوه برای افراد حرفه‌ای که آشنایی کامل با شیوه کامپایل کد منبع هستند پیشنهاد می‌شود و بیشترین کاربرد آن در بستر توزیع‌های لینوکس است که نسخه آفلاین و کامپایل شده برای بعضی از توزیع‌های آن موجود نیست.

### دربافت کد منبع و کامپایل به صورت دستی

اندازه فایل	لینک دریافت	نسخه
۴۷۹ مگابایت	برای دریافت می‌توانید کلیک کنید	نسخه tar.xz
۷۹۳ مگابایت	برای دریافت می‌توانید کلیک کنید	نسخه zip

### روش سوم (روش پیشنهادی) Offline Installers

در این روش بدون آن که نیاز باشد شما کد منبع کتابخانه را پیکربندی و کامپایل کنید، می‌توانید تنها با در اختیار داشتن اطلاعات پلتفرم خودتان و همچنین هدفی که دارید آن را دریافت و نصب کنید.

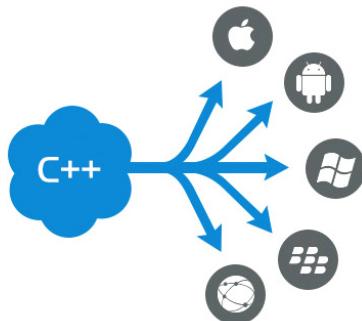
### نصب به صورت آفلاین

اندازه فایل	لینک دریافت	معماری	پلتفرم
۱.۰ گیگابایت	برای دریافت کلیک کنید	X86 - 32Bit	لينوكس
۲.۳ گیگابایت	برای دریافت کلیک کنید	X86 - 32Bit	ويندوز
۳.۵ گیگابایت	برای دریافت کلیک کنید	X64 - 64Bit	مکینتاش

### نکات بسیار مهم

- توجه داشته باشید در صورتی که شما نیاز به تولید برنامه‌های دسکتاپی برای PC و لپتاپ هستید نسخه‌ای را دریافت کنید که در بخش معماری جدول به عنوان Desktop اشاره شده است که با توجه به نوع سیستم‌عامل می‌توانید آن را تهیه کنید.
- در نسخه ۵.۹ کیوت به بعد پکیج‌های موبایل و دسکتاپ در یک بسته کامل ارائه می‌شوند.
- جهت دسترسی به نسخه iOS حتما باید از پکیج مخصوص macOS استفاده کنید که همراه با نسخه اندروید ارائه می‌شود.
- در کیوت ۵.۹ به بعد نسخه‌های دسکتاپ ویندوز مدرن تحت فناوری Universal و WinRT برای ویندوز ۱۰ و ویندوز فون مخصوص دستگاه‌های موبایلی همه در یک بسته کامل ارائه می‌شوند.
- قبل از اینکه اقدام به دریافت نسخه مورد نظر خود کنید از معماری پلتفرم خود مطمئن شوید برای مثال در بخش معماری جدول به گزینه‌های ۳۲ و ۶۴ بیت اشاره شده است که معمولاً در پلتفرم ویندوز و لینوكس این دو گزینه موجود هستند و در پلتفرم مکینتاش به طور کلی از معماری ۶۴ بیتی به صورت پیشفرض پشتیبانی می‌شود.
- لینک‌های مربوط به دریافت کتاب و فایل‌ها بر روی سرور شرکت ارائه شده‌اند.

توجه : نیازی نیست شما برای هر کدام از پلتفرم‌ها بازنویسی مجدد و انجام دهید، شما می‌توانید برای شروع برای ویندوز یا لینوکس یک نسخهٔ مورد نظر را دریافت و برنامه خود را بسازید. در نهایت که اگر نیازی برای کامپایل بر روی سیستم‌های دیگری مانند iOS و Mac یا Android باشد تنها کافی است نسخهٔ مربوط به آن سیستم‌عامل را دریافت و توسط آن پروژه خود را Import و سپس مجددآ آن را کامپایل کنید.



طراحی، کدنویسی، اشکال‌زدایی و گسترش سریع

## معرفی محیط توسعه Qt Creator نسخه ۴

محیط‌های توسعه مختلفی برای (بستهٔ ابزار) تولکیت Qt وجود دارد که اکثرًا توسط برنامه‌نویسان علاقه‌مند به این تولکیت ایجاد شده‌اند. جدیدترین محیط توسعه‌این تولکیت **Qt Creator** نام دارد. این محیط همراه با نرم‌افزارهای دیگری برای راحتی کار با لینوکس عرضه می‌شود از جمله Qt Assistant که مجموعه از کاربردها و طرز استفاده از کتابخانه‌های کیوت، Examples and Demo که مثال‌هایی برای آشنایی هر چه تمام کیوت استو زبان‌شناسی که برنامه‌ای به منظور ترجمه نرم‌افزارهای نوشته شده به این زبان و یا ساخت نرم‌افزارهای چند زبانه با سادگی هر چه تمام تر است و مهم‌تر از همه Qt Designer که نرم‌افزاری برای طراحی رابطه‌های کاربری با استفاده از ویجت و فناوری کیوت کوئیک (دکمه، کادر متن و...) های از پیش طراحی شده است. همچنین توسط افزونه‌هایی که برای این کتابخانه نوشته شده است شما می‌توانید از محیط‌هایی مانند Visual Studio نیز استفاده نمایید.

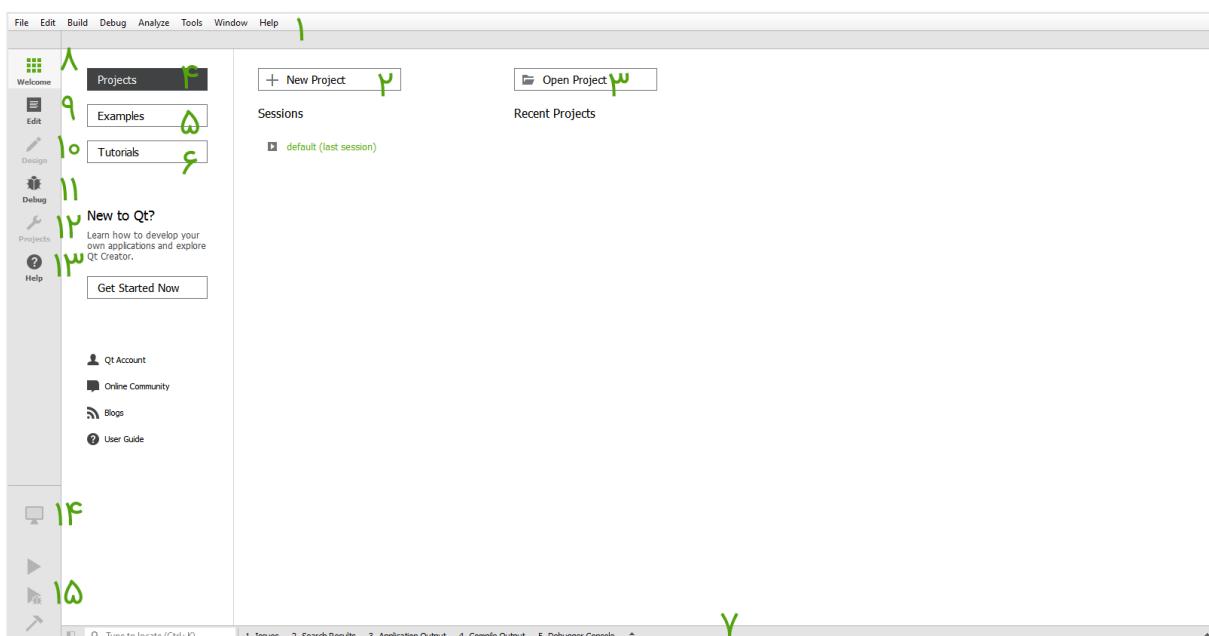
مهم‌ترین ویژگی این ابزار این است که برنامه‌های نوشته شده با این محیط توسعه، قابلیت اجرا و پشتیبانی در طیف وسیعی از سیستم‌عامل‌ها نظیر ویندوز، لینوکس، مک و حتی تلفن‌های همراه نظری سیستم‌عامل سیمی‌بین را داراست که در کنار همه این‌ها از نکات مهم این نرم‌افزار پشتیبانی از زبان‌های برنامه‌نویسی نظیر QML, CSS & JavaScript و غیره... است.

خلاصه‌ای از ویژگی‌های این محیط به صورت زیر است:

- پشتیبانی از تمامی سیستم‌عامل‌های رایج مانند ویندوز، مک، لینوکس و ...
- امکان ایجاد انواع پروژه‌های Qt Quick, Qt Canvas 3D, QWidget و ...
- پشتیبانی بسیار عالی از فناوری‌های Qt Quick برای توسعه سریع UX و UI تحت زبان QML در C++
- سیستم گزارش باگ و اشکال زدایی پیشرفته و کارآمد
- آنالیز کردن کدهای نوشته شده
- CVS, SubVersion, Mercurial, Bazaar
- مدیریت Git

- اجرای تست دستی، توانایی تست خودکار UI در برنامه‌های Cross-Platform
- اجرای تست عملکرد (Performance Testing)، تست بارگذاری (Load Testing)
- امکان آنالیز و خطایابی کد به صورت پیشرفته
- نوآوری‌های و امکانات جدید در زبان‌های تمامی زبان‌های پشتیبانی شده
- پشتیبانی از تقریباً تمامی کتابخانه‌های موجود در C++
- دارای مستندات و کتابخانه عظیمی از آموزش‌های رایج و دستور العمل‌های C++ و QML
- دارای بانک عظیمی از مثال‌ها و نمونه برنامه‌های نوشته شده تحت C++ و فناوری‌های مربوط به آن
- امکان ادغام پروژه‌های C++ و زبان‌های برنامه‌نویسی دیگر.

قبل از آغاز کار با محیط کیوت لازم است به وظایف بخش‌های مشخص شده اشاره‌ای شود.



همانطور که مشخص است بر اساس آخرین تغییرات طبق تصویر می‌بینیم که تقریباً حدود ۱۳ بخش مختلفی وجود دارد که هر کدام به صورت زیر دارای وظایف و کاربرد هایی هستند.

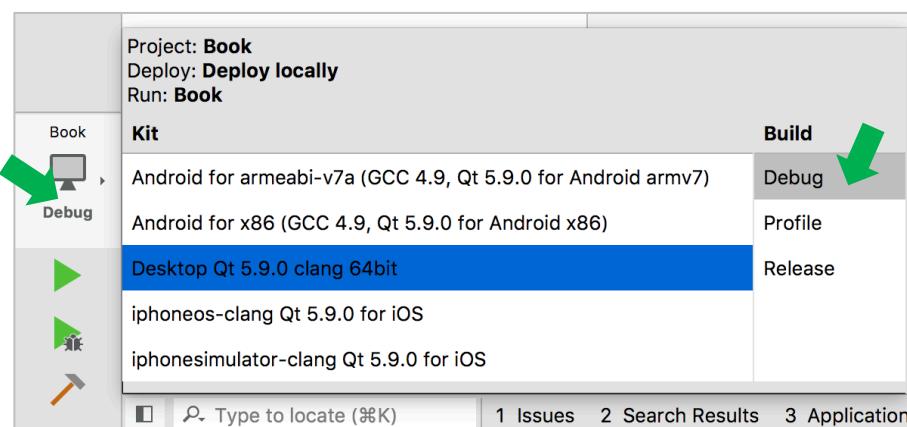
۱. منوی اصلی / نوار ابزارها (شامل منوی‌های استانداردی است که اکثراً در هر محیط برنامه‌نویسی یافت می‌شود)
۲. شامل گزینه New project جهت ایجاد پروژه جدید است.
۳. گزینه Open project جهت وارد کردن پروژه‌های موجود با پسوندهای .pro, .creator, .pyqtc, .qbs, .qmlproject, .cmakeLists.txt و همچنین اجازه دسترسی برای شروع کار طبق آخرین جلسات کاری و جلسات ذخیره شده به صورت پیشفرض را در اختیار ما قرار می‌دهد.
۴. شامل تمامی پروژه‌ها بود و در واقع محیط دسترسی به پروژه‌ها در این زبانه قرار می‌گیرد.
۵. یکی از قابلیت‌های خوب محیط توسعه Qt داشتن بخش بسیار جامعی که شامل مثال‌ها و آموزش‌های بسیار زیادی است که برای دسترسی به آن‌ها در این بخش یعنی زبانه Examples می‌تواند اقدام کرد.
۶. بخش آموزش‌های آنلاین و تمامی سeminارها و ویدیوهای آموزشی رسمی توسط کیوت در این بخش ارائه می‌شود.

۷. این بخش شامل نوار وضعیت‌های پروژه است که شامل موارد Output و ... برای مشاهده انواع رخدادها و خروجی‌ها به صورت کنسول است. در این بخش می‌توانید در سریع‌ترین زمان ممکن به وضعیت پروژه در زمان کامپایل و اجرا دسترسی داشته باشید.
۸. زبانهُ اصلی که شامل موارد ۲ تا ۷ است.
۹. قسمت ویرایش یا همان Edit بخش ویرایش که در صورت وجود پروژه شامل محتوای پروژه شما است و آن را به صورت ساختار درختی نمایش می‌دهد که در واقع امکان انتخاب فایل‌های موجود در پروژه را نیز می‌دهد.
۱۰. بخش طراحی و Design شامل محیط طراحی، ابزارها و موارد مرتبط با طراحی پروژه است.
۱۱. بخش Debug مربوط به عملیات دیباگینگ (اشکال زدایی) است.
۱۲. این بخش مربوط به تنظیمات پروژه مانند تنظیمات نوع سازنده، کامپایلر و غیره است.
۱۳. یکی از بهترین قسمت‌های پرکاربرد برای مبتدیان بخش Help خواهد بود که می‌توانید با مراجعه با این قسمت و جستجوی دستورات و عبارات مورد نظر آموزش‌هایی را در رابطه با موضوع مورد نظر را دریافت نمایید.
۱۴. اجرای برنامه همراه با کامپایل، در صورتی که برنامه ساخت شده باشد برنامه فقط اجرا خواهد شد در غیر اینصورت بر اساس تنظیمات پیش فرض کامپایل و سپس اجرا خواهد شد.
۱۵. نوع کامپایل و خروجی را تنظیم می‌کند و عملیاتی مانند: Build، Rebuild، Compile، Run و حتی Profiler و ... را انجام می‌دهد.

### پیکربندی و تنظیمات مربوط به ساخت برای پلتفرم‌های مختلف

در محیط توسعه کیوت امکان تنظیم کیت مخصوص پلتفرم مورد نظر شما میسر است. همچنین انتخاب نوع کامپایلر و یا دیباگر در آن فراهم شده است که بر اساس نیاز از آن‌ها استفاده شود. که بعد از ایجاد پروژه به طور پیش‌فرض برنامه شما برای محیطی که می‌خواهید منتشرش کنید آماده سازی می‌شود. در صورتی که در کیت مشکلی موجود باشد می‌توانید در بخش Issues آن را مشاهده و بر اساس اطلاعات دریافتی اقدام به حل آن نمایید.

برای سایت و کامپایل پروژه کافی است بر روی آیکون کیت که در تصویر زیر مشخص شده است کلیک کنید تا کیت‌های موجود بر روی محیط توسعه نمایان شوند.



سپس در بخش Kit نوع کامپایل را بر اساس نوع پلتفرم مقصد مشخص کرده و در بخش Build نوع کامپایل را مشخص می‌کنیم. در این قسمت می‌توانید در سه حالت Release, Debug, Profile را تنظیم نمایید که به طور پیش‌فرض موجود است. در این بخش با انتخاب نوع کیت و نوع (حالت) کامپایل شدن برنامه پروژه بر این اساس تنظیم و پیکربندی خواهد شد. لذا در صورتی که نیاز باشد تمامی اثرات مربوط به این تنظیمات را نسبت به کیت حذف نمایید کافی است از منوی CleanAll گزینه Build را انتخاب کنید. این کار باعث می‌شود پوشه مربوط به دایرکتوری (مسیر) ساخت پروژه را پاکسازی و آماده برای انتشار نماید.

حال برای ساخت (Build) پروژه برای دریافت خروجی از آن کافی است از منوی Rebuild All گزینه Build را انتخاب کنید. در نهایت بعد از ساخت پروژه فایل‌های مورد نیاز در مسیر مربوطه ساخته خواهند شد.

**نکته:** برای ساخت و انتشار پروژه بعد از تغییرات در کدها و منابع از Clean All و Rebuild All استفاده کنید.

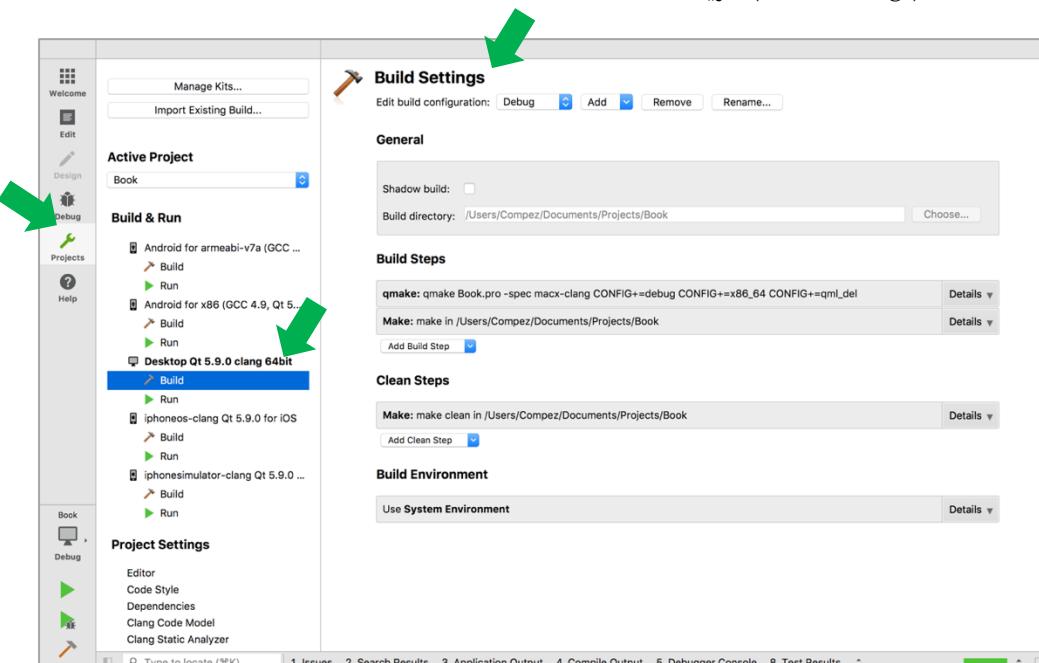
به طور عادی این مراحل به صورت پیشفرض تحت محیط توسعه در اختیار شما قرار می‌گیرد، ممکن است نیاز باشد پروژه را بر اساس سلیقه شخصی در مسیر و پوشه مورد نظر ایجاد و منتشر سازیم. مسیر انتشار (ساخت) فایل‌های خروجی تحت عنوان **Shadow Build** در کیوت کریتور مشخص می‌شود که به معنی این است (پروژه شما در یک دایرکتوری جداگانه ساخته خواهد شد) که دایرکتوری متفاوت و جدا از دایرکتوری منبع سورس کدهای مرتبط با پروژه است. در کل استفاده از گزینه Shadow Build به خاطر این است که دایرکتوری منبع اصلی پروژه شما تمیز و جدا نگهداری شود. این باعث می‌شود شما سریعتر بین تنظیمات پروژه بر اساس نوع تنظیمات خود سوئیچ کنید. چرا در بحث چند-سکویی (Cross-Platform) این گزینه بسیار کارآمد است.

در صورتی که گزینه **Shadow Build** از حالت انتخاب در آورده شود تمامی فایل‌های ساخته شده در دایرکتوری پیشفرض داخل پروژه در کنار منبع کد اصلی ساخته خواهد شد.

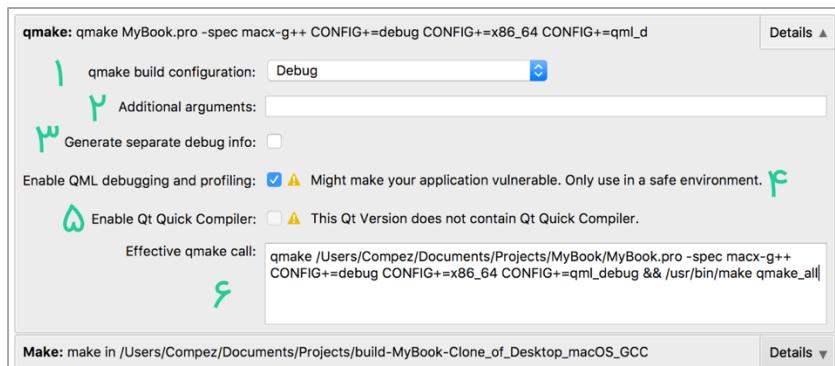
برای مثال: اگر نام پروژه ما MyProject باشد، و گزینه Shadow Build فعال باشد. در زمان ساخت، خروجی‌های تولید شده در پوشه‌ای با رشته مسیر زیر ساخته خواهد شد:

..../build-MyProject-نوع کیت-کامپایل

در غیر این صورت در داخل پوشه اصلی پروژه یعنی .../MyProject/.. تولید و ذخیره سازی خواهد شد. در این صورت کافی است در قسمت Projects وارد تنظیمات ساخت (Build Settings) شوید.



محیط کیوت کریتور وظیفه ساخت و اجرای پروژه را در اختیار دارد که برای اجرای آن از فرامین (شیل) که بر روی سیستم به طور بومی هستند استفاده می‌کند، که تحت ابزارهای make و qmake اجرا می‌شوند. برای مشخص کردن مسیر مورد نظر برای تولید فایل‌های خروجی کافی است مقدار Build Directory را بر اساس سلیقه خودتان مشخص نمایید.



کافی است بر روی گزینه Details ابزار مربوطه کلیک کنید جزئیات تنظیمات آن به صورت تصویر زیر نمایان خواهد شد که در آن تنظیمات سریع در رابطه با نوع کامپایل، آرگومان‌های ورودی و غیره قابل دسترس هستند.

۱. حالت کامپایل پروژه در این بخش مشخص می‌شود که شامل دو گزینه Release و Debug است.
۲. آرگومان‌های اضافی و مورد نیاز در این بخش وارد می‌شوند.
۳. برای تولید و ساخت علائم اشکال‌زدایی در برنامه که در حالت Release است مورد استفاده قرار می‌گیرد. با انتخاب این گزینه می‌توان از قابلیت CPU Usage Analyzer در محیط توسعه کیوت کریتور بهره‌مند شد.
۴. انتخاب این گزینه باعث می‌شود تا بتوانید کدهای نوشته شده در استناد QML ای را مورد اشکال زدایی قرار دهید. در صورتی که این گزینه غیرفعال باشد اگر اشکالی در کدهای نوشته شده تحت QML رخ دهد ممکن است از آن باخبر نشود.
۵. برای کامپایل سورس کدهای مربوط به QML از این گزینه استفاده می‌شود. این قابلیت باعث بهبود زمان اجرای پروژه شده و تمامی فایل‌های مورد نیاز جهت استفاده در کنار فایل پروژه را از بین می‌برد **که تنها در نسخه تجاری Qt قابل استفاده است.**
۶. تمامی تنظیمات مربوط به (پرچم - flag) هایی که تحت ابزار qmake برای ساخت پروژه در نظر گرفته شده است در این قسمت قابل مشاهده و بررسی می‌باشند.

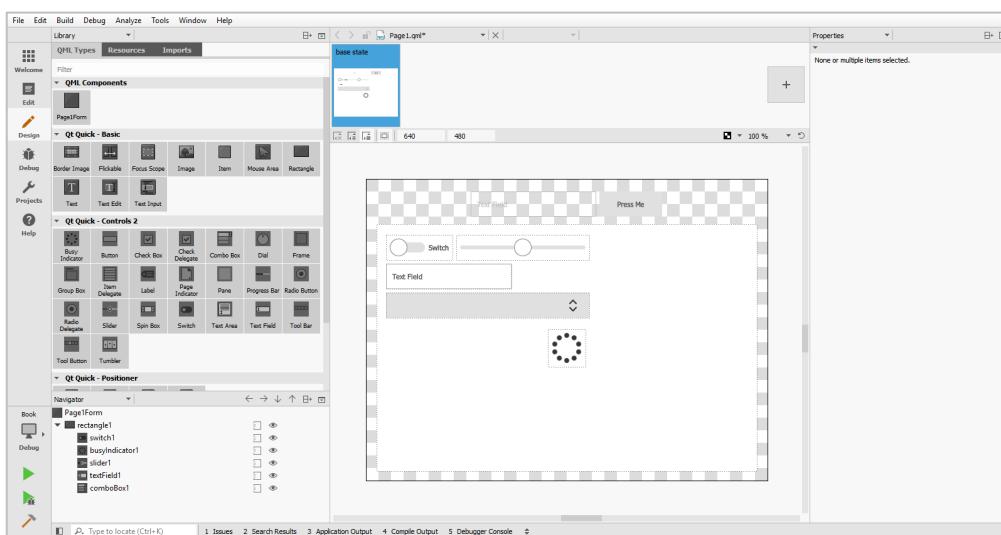
در بخش Build Environment امکان دسترسی به متغیرهای محیطی (Environment Variable) فراهم می‌شود که به مجموعه‌ای از مقادیر نام‌گذاری شده که می‌توانند نحوه رفتار کردن فرآیندهای در حال اجرا را تغییر داده و بر روی آنها اثر بگذارند در اختیار قرار می‌دهند. برای استفاده از متغیرهای محیطی می‌توان از این قسمت استفاده کرد.

کافی است بر روی گزینه Build Environment رفته و سپس گزینه Details زبانه Build Settings وارد شوید. متغیرهای محیطی با استفاده از ترکیب بومی محیط توسعه قابل استفاده هستند برای مثال \${VARNAME} یا \$VARNAME مختص ایستگاه‌های یونیکس و %VARNAME% مختص ایستگاه‌های ویندوز می‌باشند.

همچنین علاوه بر این می‌توان از متغیرهای محیطی توسعه کیوت نیز استفاده کرد. برای مثال متغیر **HOME** را می‌توان مقدار دهی کرد که ارزش آن برابر است با `/Users/YourComputerUserName`.

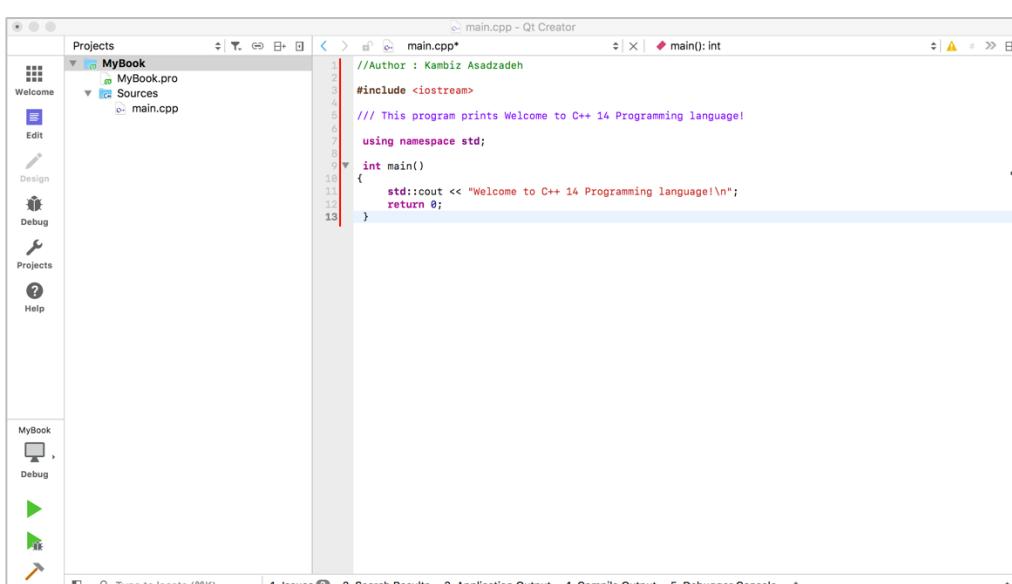
## بخشی از ویژگی‌های محیط توسعه

پوسته‌های Flat-Light (روشن-تخت) و Flat-Dark (تاریک-تخت) در نسخه ۴ به این محیط توسعه افزوده شده که در همین نسخه نیز بهبود یافته است و از طریق مسیر Environment > Interface > Theme settings.



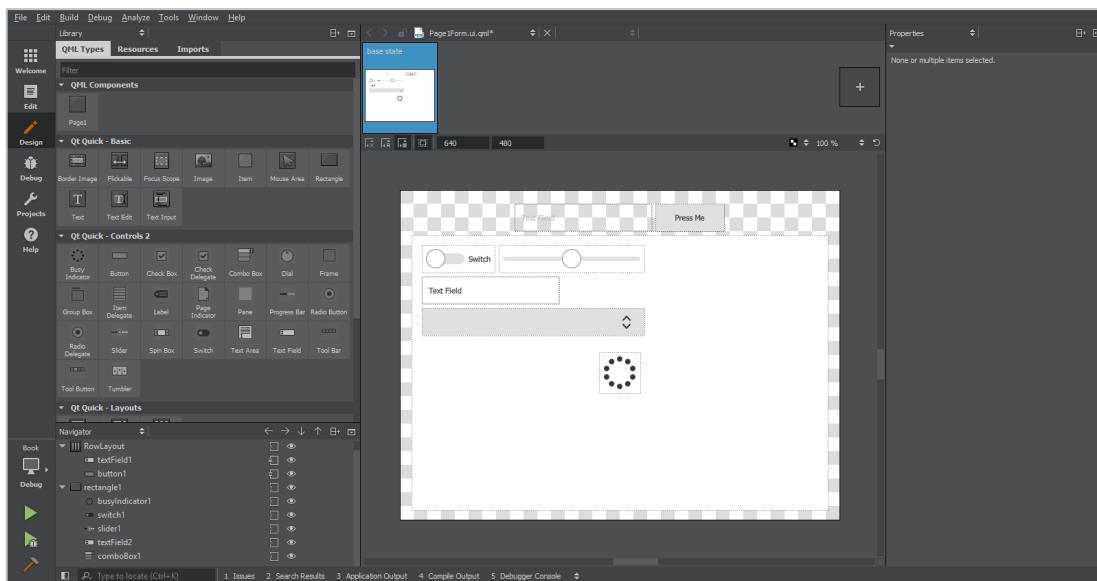
## بخش ویرایش

ویرایشگر در حال حاضر بسیار بهتر و هوشمند تر شده است و به خوبی می‌تواند به صورت خودکار گزینه‌های مورد نظر را تشخیص و یا جایگزین نماید. برای مثال اگر شما کاراکتر نقل قول ("") را ایجاد کنید در ادامه آن به صورت خودکار خاتمه دهنده اضافه خواهد شد. اگر شما کاراکتر مربوط به باز شدن و بسته شدن را تایپ کنید به صورت هوشمند قابل تشخیص خواهد بود. برای تنظیمات می‌توانید از منوی Completion و Text Editor به اقدام کنید.



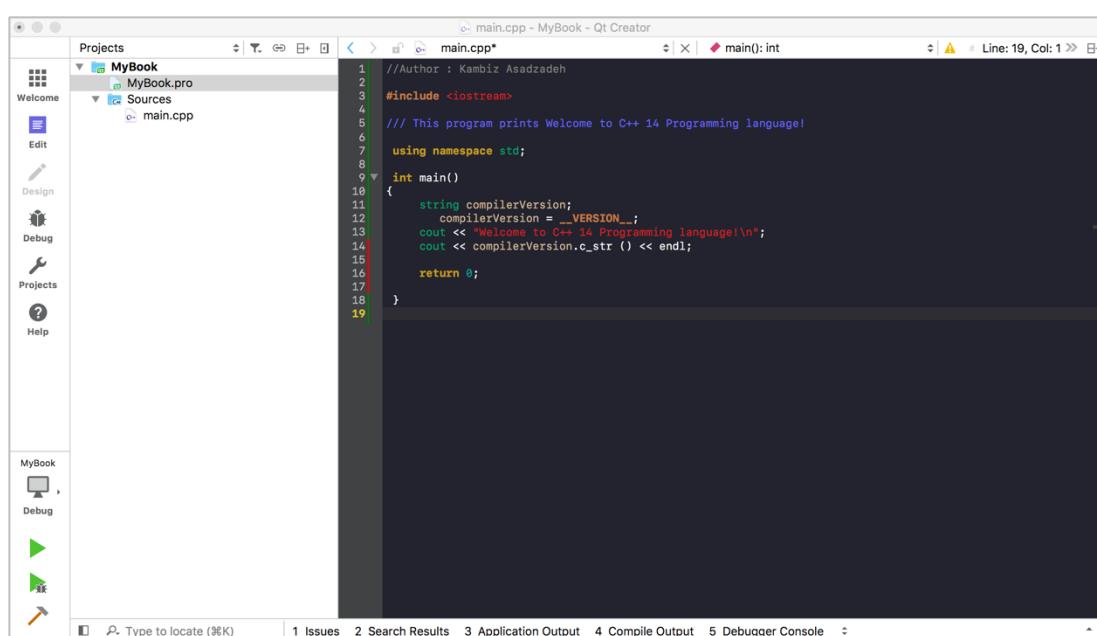
## پشتیبانی از Qt Quick

هر دو گزینه QML Profiler و Qt Designer در بهبودهای بسیاری را در عملکرد دریافت کرده‌اند. هم اکنون شما می‌توانید یک سبک Qt Quick در بهبودهای بسیاری را در Qt Quick Designer به آن در Qt Quick Controls 2 انتخاب و از اقلام و آیتم‌های مربوط به آن در Qt Quick Designer برای تولید آیتم‌های خود استفاده کنید.



## پشتیبانی از C++

گذشته از رفع اشکالات در Code Model و Static Analyzer بسته‌های باینری برای استفاده Clang و همچنین رفع بسیاری از باگ‌های مرتبط با صورت گرفته است. همچنین پچ‌های اخیر برای کارکرد بهتر در Clang و نسخه به روز رسانی شده MSVC 2017 اضافه شده است.



سی‌میک یکی از بهترین و معروف‌ترین ابزارهای آزاد و متن‌باز برای مدیریت فرآیند ساخت (ساخت) نرم‌افزار با استفاده از شیوه‌ای مستقل از کامپایلر است که ساختارهای پوشاهی و برنامه‌هایی که به چندین کتابخانه وابسته‌اند را حمایت می‌کند. در کیوت این ابزار در کنار ابزار اختصاصی کیو-میک (qmake) وجود دارد و در صورتی که ابزار cMake را بر روی سیستم خود نصب کرده باشید می‌توانید از آن بهره‌مند شوید.

همچنین در نسخه ۵.۸ به بعد کیوت می‌توانید متغیر **QML\_IMPORT\_PATH** را در مخزن CMake پروژه خودتان وارد کنید. Qt Creator آن را شناسایی و برای استفاده در QML اعمال می‌کند. بنابراین شما می‌توانید هم اکنون با استفاده از واردات QML در ویرایشگر خود بهره‌مند شوید. به عنوان مثال این کد را درج کنید:

```
(QML_IMPORT_PATH ${CMAKE_SOURCE_DIR}/qml ${CMAKE_BINARY_DIR}/imports CACHE string "" FORCE))
```

## معرفی مجوزهای کیوت (Qt) و نحوه استفاده از مناسب‌ترین مجوز (License)

کیوت (Qt) در دو نسخه منبع باز و تجاری عرضه می‌شود، نسخه تجاری آزاد برای هر نوع توسعه نرم‌افزار است. در حالی که نسخه منبع باز محدود به توسعه نرم‌افزار در قالب مجوزهای جی‌پی‌ال (GPL) و کیو‌پی‌ال (QPL) است. البته از نسخه ۴,۵ به بعد امکان توسعه نرم‌افزار در قالب مجوز ال‌جی‌پی‌ال (LGPL) نیز به مجوزهای نسخه منبع باز اضافه شده است همچنین بسیاری از مازول‌ها و قابلیت‌هایی که در نسخه پنجم تحت مجوز تجاری در دسترس بودند خوشبختانه در نسخه ۵.۷ به بعد رایگان و تحت مجوز ال‌جی‌پی‌ال - ویرایش سوم (LGPLv3) عرضه شده‌اند.

نکته: توضیحات مربوط در زمان تالیف این کتاب در نسخه ۵.۹ و ۵.۱۳.۱ است، بنابراین ممکن است در نسخه‌های بعدی شرایط مرتبط به مجوزها تغییر پیدا کند.

### برخی از قابلیت‌های تجاری موجود هستند عبارتند از:

صفحه کلید مجازی (Qt Charts)، نمودارها (Qt Virtual Keyboard) و غیره... که در نسخه‌های جدید ۵.۸ تحت لیسانس LGPL منتشر شده‌اند. بنابراین شاید سوال پیش آید که تفاوت بین نسخه‌های تجاری و رایگان در چه چیزی است. در زیر جدول مقایسه انواع مجوزها را مشاهده می‌کنید که تفاوت بین نسخه‌های Commercial، GPL و LGPL (تجاری) در آن مشخص شده است.

(GPLV2/GPLV3)	(LGPLV3)	(Commercial)	نوع مجوز
رایگان	رایگان	شروع از ۷۹ دلار در ماه	قیمت
خدمات و حقوق مربوطه			
✓	✓	✓	پشتیبانی جامعه برنامه‌نویسان
X	X	✓ **	پشتیبانی رسمی دیجیا (Digia)
X	X	✓	نگه داری برنامه به صورت شخصی
X	✓	✓	حفظ و نگه داری سورس برنامه در حالت لینک داینامیکی
X	X	✓	نیاز نداشتن به مکانیزم لینک مجدد در صورتی که لینک استاتیکی نباشد
X	X	✓	عدم ارائه لاینس کیوت و عدم قید استفاده از کتابخانه کیوت

X	X	✓	عدم ارائه نسخه‌ای از سورس کیوت به مشتریان
X	X	✓	حق دسترسی کامل برای ویرایش سورس کد کتابخانه کیوت
(GPLV2/GPLV3)	( <a href="#">GPLV3</a> )	(Commercial)	تجاری
✓	✓	✓	ماژول‌ها
✓	✓	✓	ماژول Qt Essentials
✓	✓	✓	ماژول‌های اضافی کیوت
✓	✓	✓	ماژول Qt Charts
✓	✓	✓	ماژول Qt Data Visualization

(GPLV2/GPLV3)	( <a href="#">GPLV3</a> )	(Commercial)	ابزارها
✓	✓ *	✓	محیط توسعه Qt Creator با قابلیت‌های عمومی
✓	✓ *	✓	ابزارهای مستند و کتاب‌ها
✓	✓ *	✓	ابزارهای درونی سازی
✓	✓ *	✓	محیط توسعه Qt Quick
✓	✓ *	✓	امکان استفاده از Qt Quick Profiler
✓	✓ *	✓	ابزار فراهم کننده استفاده از Qt Quick در Visual Studio
✓	X	✓	تولید تصاویر توسط OpenGL درین دخالت Qt Quick
X	X	✓	کامپایلر Qt Quick
X	✓	✓	Qt Virtual Keyboard
X	X	✓	امکان دیباگینگ (عیب‌یابی) مستقیم بر روی دستگاه
X	X	✓	دریافت خروجی مناسب با تنها یک کلیک بر روی دستگاه
X	X	✓	ابزار Boot to Qt پیش‌سازنده در امتداد تحت لینوکس
X	✓	✓	دستورالعمل‌های پروژه Yocoto برای تصویر سفارشی
X	✓	✓	به روز رسانی‌های مرتبط با پروژه Over-the-Air

\* ابزارهایی که با مجوز GPLv3 عرضه شده‌اند می‌توانند با پذیرش شرایط GPLv3 مورد استفاده قرار بگیرند به شرطی که تغییری در منبع سورس کد اصلی محصول اعمال نشود و یا تغییرات منتشر شود. \*\* پشتیبانی شامل طرح استارت-آپ نمی‌شود.

طبق توضیحات زیر و با توجه به جدول مجوزها شما می‌توانید برنامه خود را به صورت زیر توسعه و منتشر نمایید.

### در [Mجوز](#) [GPL](#)

- شما می‌توانید برنامه خود را نوشه و توسعه دهید همچنین حق آن را دارید که سورس آنها را منتشر نکنید.
- می‌توانید برنامه‌هایی بنویسید که با مجوز ([Genyleap LLC](#)) می‌توانید آنها را منتشر نمایید.

مجوز تحت حق چاپ شرکت / کامبیز اسدزاده با این برچسب می‌تواند روی محصول ارائه گردد. این بدین معنی است که تمامی حقوق مربوطه متعلق به نام تجاری شما، شخصیت حقوقی یا حقیقی شما است.

- باید قید کنید برنامه شما از کیوت استفاده کرده است همچنین سازندگان کیوت توصیه می‌کنند از بنر «ساخته شده با کیوت» استفاده کنید (این یک شیوه مناسب برای حمایت از کتابخانه محبوبتان است) و می‌توانید برچسب مربوطه را دریافت و استفاده کنید.
- باید امکانی برای کاربر فراهم کنید تا به سورس کیوت دسترسی داشته باشد (ساده‌ترین راه معرفی سایت رسمی کیوت است). تبلیغات و معرفی کتاب خانه و زبان برنامه‌نویسی بر روی یک پروژه می‌تواند کاربر را مطمئن سازد.
- اگر تغییری در منبع کدهای کیوت اعمال کنید باید تغییرات را نیز منتشر و اطلاع رسانی کنید.
- در صورتی که از شیوه کامپایل استاتیک استفاده کنید برنامه شما باید با مجوز سازگار با گنو عرضه شود.

### در مجوز GPL

- برنامه شما حتماً باید منبع باز بوده و با مجوز سازگار با گنو ارائه شود.
- باید قید کنید برنامه شما از کیوت استفاده کرده است. در صورتی که این مطلب را جایی عنوان نکنید به دلیل منبع باز بودن برنامه و اینکه استفاده از کیوت قطعی است عمل خلافی صورت نگرفته است.
- اگر تغییری در منبع کیوت اعمال کنید باید تغییرات را نیز منتشر کنید.
- بر خلاف LGPL از روش کامپایل استاتیک (ایستا) نمی‌توانید استفاده کنید.

### در مجوز تجاری (Commercial)

- در صورتی که تمایل دارید از این نسخه استفاده کنید باید هزینه مربوط به آن را پرداخت کنید و پس از آن می‌توانید هر آنچه را که می‌خواهید انجام دهید.

### قید استفاده از کیوت و ارجاع به منبع کیوت

برای آنکه عنوان کنید از کیوت استفاده کرده‌اید راه مشخصی وجود ندارد، تنها فرمول آن این است که در جایی که کاربر می‌تواند ببینید توضیحی ارائه کنید و لینک به سایت کیوت عرضه کنید. این محل ممکن است یک فایل راهنمایی (Readme) و یا صفحه توضیحات در Google Play Store باشد.

همچنین شما می‌توانید در صفحه درباره محصول یا درباره ما (About) این مطلب را عنوان کنید و یک کپی از موافقتنامه GPL یا LGPL را نیز به کاربر نشان دهید و عنوان کنید این قانون کیوت است. به عنوان یک مثال در این زمینه به قسمت About تلگرام (Telegram) مراجعه کنید.

### لوگوهای نشانگر ساخته شده با Qt

برای برنامه‌های نوشته شده با کیوت می‌توانید از لوگوهایی که تیم کیوت آماده کرده است استفاده کنید. این لوگوها بیان کننده این است که برنامه شما با کیوت توسعه یافته است. استفاده از این لوگوها فراتر از عمل به قانون است، درواقع راهی است تا به عنوان یک عضو جامعه برنامه‌نویسان

کیوت از آن حمایت کنید. کیوت در این باره ۶ نوع لوگو در نظر گرفته است که البته باید توجه داشته باشید که استفاده غیر مجاز از لوگو می‌تواند کبی رایت و سایر قوانین را نقص کند.



جهت دریافت لوگوهای مربوطه می‌توانید به آدرس <https://iostream.ir/library/gui/qt.html> مراجعه نمایید.

## پشتیبانی از انواع پلتفرم‌ها

به طور طبیعی برنامه‌نویسی با C++ در تمامی سیستم‌عامل‌ها و تجهیزات امکان‌پذیر است، حال که برای طراحی و پیاده‌سازی یک برنامه در محیط Qt فراهم شده است به قابلیت‌های این کتابخانه در این زمینه می‌پردازیم.

باید بگوییم با حداقل بازنویسی و طراحی مجدد شما می‌توانید برنامه‌ای را که طراحی کرده‌اید در نسخه‌های مختلف دسکتاپ، موبایل و سیستم‌های جدا از آن‌ها اجرا کنید و این یکی از بهترین امتیازها برای یک برنامه‌نویس است که یک بار کد نوشته شده را در سیستم‌های مختلف بدون بازنویسی مورد استفاده قرار داده و در صورت لزوم آن را کامپایل کند. در کل به خاصیت و قابلیت اجرا شدن کد بر روی سکوهای مختلف حالت چند-سکویی می‌گویند.

## پشتیبانی از انواع معماری‌ها

معماری‌های ۳۲ و ۶۴ بیت به صورت کامل در این کتابخانه پشتیبانی می‌شود که معمولاً در هر پلتفرم تحت GCC و دیگر کامپالی‌ها می‌توان آن را ساخت. باید به این نکته اشاره کنیم که قابلیت پشتیبانی از OpenGL (ES) 2.0, DirectX 9 و جدیداً Vulkan یا یک تولید کننده تصویر دیگر در فناوری Qt Quick پشتیبانی می‌شود که علاوه بر آن در حالت Widgets استفاده از شتاب دهنده‌ها سخت افزاری نیاز نیست.

**پلتفرم‌هایی که پشتیبانی می‌شوند شامل موارد زیر هستند:**

**نسخه‌های دسکتاپی :**

- مکیناچ (macOS)
- ویندوز (Windows)
- لینوکس (Linux)

**: (Embedded) نسخه‌های تعییه شده**

- ویندوز (Windows)
- لینوکس (Linux) تمامی سیستم‌عامل‌های بر پایه Unix

• آندروید (Android)

• کیو ان ایکس (QNX)

• وی ایکس ورکس (VxWorks)

### نسخه های موبایلی :

• ویندوز فون (Windows Phone)

• آی او اس (iOS)

• اندروید (Android)

• بلک بربی (Blackberry)

\* از دیگر سیستم عامل ها می توان به Tizen و sailfishos هم اشاره کرد.

در زیر جدولی از انواع سیستم عامل ها و کامپایلرهای سازگار با این کتابخانه موجود:

یادداشت / نکته	کامپایلر	پلتفرم
<b>پلتفرم ویندوز</b>		
	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز ۱۰ (۶۴-بیت)
	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز ۱۰ (۳۲-بیت)
	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز 8.1 (۶۴-بیت)
	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز 8.1 (۳۲-بیت)
	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز 7 (۶۴-بیت)
کامپایلر MinGW تحت ساخته می شود. 5.3	MSVC 2017, MSVC 2015, MSVC 2013, MinGW 5.3	ویندوز 7 (۳۲-بیت)
<b>پلتفرم لینوکس</b>		
	GCC 4.8.5	توزیع (۶۴-بیت) openSUSE 42.1
devtoolset-4	GCC 4.9.5	توزیع (۶۴-بیت) Red Hat Enterprise Linux 6.6
devtoolset-4	GCC 5.3.1	توزیع (۶۴-بیت) Red Hat Enterprise Linux 7.2
	کامپایلر GCC ارائه شده توسط کانونیکال	توزیع (۶۴-بیت) Ubuntu 16.04
	GCC 4.8, GCC 4.9, GCC 5.3	توزیع (۳۲ و ۶۴ بیتی) لینوکس
<b>پلتفرم مکینتاش</b>		

	پشتیبانی تحت کامپایلر کلنگ (Clang) اپل	نسخه 10.10, 10.11, 10.12
<b>پلتفرم‌های جا سازی شده / Embedded Linux, QNX</b>		
بوردهای ARM Cortex-A GCC بر پایه Intel	GCC	Embedded Linux
میزبانی در اوبونتو ۶۴ بیتی و ویندوز	ارائه شده توسط QNX	QNX 6.6.0, 7.0 (armv7le and x86)
<b>پلتفرم‌های موبایل / Android, iOS, WinRT</b>		
میزبانی در ویندوز ۱۰	MSVC 2017, MSVC 2015	توزیع Windows Platform (UWP)
میزبانی در سیستم عامل OS X	تحت کامپایلر کلنگ (Clang) ارائه شده توسط اپل	iOS 8, 9, 10 (armv7, arm64)
میزبانی در اوبونتو ۱۴/۰۴ (۶۴ بیت)، ویندوز و Mac OS X	تحت کامپایلر GCC ارائه شده توسط گوگل	سطح رابط ۱۶ و بالاتر Android

❖ این لیست ممکن است در ویرایش‌های آتی تغییر کند.

## شرایط و قوانین لازم جهت انتشار اپلیکیشن در فروشگاه‌های مربوطه

یکی از سوالات مهم علاقه‌مندان در رابطه نحوه انتشار اپلیکیشن‌های نوشته شده تحت کیوت در فروشگاه‌های رسمی پلتفرم‌ها این است که روش و قوانین موجود در این زمینه چگونه است. قبل از اینکه ما به قوانین موجود در کیوت اشاره کنیم نیاز است در نظر داشته باشیم که فروشگاه‌های مربوطه خود نیز دارای قوانینی هستند که توسعه‌دهنده موظف است آن را رعایت کند. برای مثال تمامی قوانین موجود در فروشگاه‌ها در لیست زیر برای تمامی برنامه‌نویسان سی‌پلاس‌پلاس و غیر این زبان شامل می‌شود.

### • شرایط موجود در Apple Store یا همان (iTunes)

- توسعه‌دهنده از طرف شرکت اپل دارای گواهی‌نامه‌ای به عنوان حق توسعه‌دهندگی خواهد بود که برای پذیرش شخص ثالث از طرف اپل به عنوان یک توسعه‌دهنده رسمی می‌باشد مبلغ ۹۹ دلار را به مدت ۱ سال حق توسعه‌دهندگی بپردازد. همچنین اگر توسعه‌دهنده به عنوان یک شخصیت حقوقی داری چندین توسعه‌دهنده است باید هزینه ۵۹۹ دلار را بپردازد تا در قبال آن حساب **Developer ID** را دریافت نماید. در صورتی که این مورد اقدام نشود به هیچ عنوان قادر به ارسال فایل برنامه با پسوند ipa بر روی آپاستور نخواهد بود.
- بعد از اینکه از رسمی شدن اکانت توسعه‌دهندگی خود مطمئن شدید می‌توانید وارد (iTunes Connect) با آدرس <http://developer.apple.com/account> شوید. این بخش کنترل پنلی است جهت ورود و مدیریت برنامه‌های شما در اپ استور.
- در صورتی که حساب شما اعتباری نداشته باشد برنامه شما تنها بر روی گوشی‌های چیل‌بریک شده (Jail Break) قابل اجرا خواهد بود.

## • شرایط موجود در Windows Store یا همان (Store)

- توسعه‌دهنده از طرف شرکت مایکروسافت دارای گواهی‌نامه‌ای به عنوان حق توسعه‌دهنگی خواهد بود که برای پذیرش شخص ثالث از طرف مایکروسافت به عنوان یک توسعه‌دهنده رسمی می‌بایست مبلغ ۱۹ دلار را به مدت ۱ سال حق توسعه‌دهنگی پپردازد. همچنین اگر توسعه‌دهنده به عنوان یک شخصیت حقوقی داری چندین توسعه‌دهنده است باید هزینه ۹۹ دلار را پپردازد تا در قبال آن اکانت Developer ID را دریافت نماید. در صورتی که این مورد اقدام نشود به هیچ عنوان قادر به ارسال فایل برنامه بر روی آپ استور مایکروسافت نخواهد بود.
- بعد از اینکه از رسمی شدن اکانت توسعه‌دهنگی خود مطمئن شدید، می‌توانید وارد (Register as an app developer) با آدرس <https://developer.microsoft.com/en-us/store/register> شوید. این بخش کنترل پنلی است جهت ورود و مدیریت برنامه‌های شما در استور ویندوز است.

## • شرایط موجود در Google Play

- توسعه‌دهنده از طرف شرکت گوگل دارای گواهی‌نامه‌ای به عنوان حق توسعه‌دهنگی خواهد بود که برای پذیرش شخص ثالث از طرف گوگل به عنوان یک توسعه‌دهنده رسمی می‌بایست مبلغ ۲۵ دلار را به مدت ۱ سال حق توسعه دهنگی پپردازد.
- بعد از اینکه از رسمی شدن اکانت توسعه‌دهنگی خود مطمئن شدید می‌توانید وارد (Google Play Developer Console) با آدرس <https://play.google.com/apps/publish/signup> شوید. این بخش کنترل پنلی است جهت ورود و مدیریت برنامه‌های شما در فروشگاه رسمی گوگل است.

**نکات قابل توجه:** همهٔ ما می‌دانیم که به خاطر تحریم‌های موجود هیچ یک از استورهای ذکر شده هنگام ثبت نام موقعیت ایران را در بین فیلدهای قابل انتخاب ندارند. بنابراین در وارد کردن و شناسایی موقعیت توسعه‌دهنده بسیار حساس عمل می‌کنند و متأسفانه باید گفت هیچ راه حلی به جز داشتن حساب رسمی با موقعیت خارج از کشور را نداریم. تنها روش قابل پرداخت برای حساب توسعه دهنگی در هر سه شرکت بر اساس دلار است و به هیچ عنوان بدون داشتن ویزا کارت و حساب ارزی قادر به پرداخت و خرید حساب کاربری جهت توسعه دهنگی نخواهیم داشت. در صورتی واجب است برنامه شما در سطح جهان منتشر شود باید راهی برای خرید حساب توسعه دهنگی به صورت رسمی فراهم آورید که کمی دشوار است. بنابراین شرایط مربوطه چنین است و در صورتی که بتوانید این حساب را تهیه کنید باید هزینه‌هایی که ذکر شده است را بپردازید. در صورتی که هدف شما انتشار برنامه در فروشگاه‌های داخلی باشد می‌بایست از قوانین و شرایط آن‌ها پیروی کنید که با کمی جستجو به دست می‌آید.

## شرایط و قوانین اختصاصی برنامه تحت Qt جهت انتشار و پذیرش در فروشگاه‌های مرتبط

- برنامه نوشته شده تحت این کتابخانه می‌بایست دارای هیچ گونه API مرموزی نباشد.
- برخلاف فروشگاه‌های داخل مراکز رسمی و خارجی به هیچ عنوان پذیرنده برنامه‌های تکراری و بی ارزش نیستند.
- فیلترینگ مخصوصی جهت بررسی صحت کارکرد برنامه شما و عدم وجود مسائل امنیتی وجود دارد که در صورت مشاهده برنامه منتشر شده حذف خواهد شد.

- در فروشگاه‌های اپل استور می‌باشد مجوز تجاری کیوت را تهیه کنید.
- طراحی رابط کاربری، کیفیت کدنویسی، طراحی آیکون نهایی جهت نمایش در فروشگاه و همچنین توضیحات مرتبط باید بسیار دقیق و از کیفیت مطلوبی برخوردار باشد. به هیچ عنوان تحت هیچ شرایطی مانند فروشگاه‌های داخلی هر برنامه‌ای حق انتشار نخواهد داشت مگر اینکه برنامه شما درجه ۱ اعلام شود.

## فصل دوم

### معرفی فناوری Qt Quick ویرایش ۲

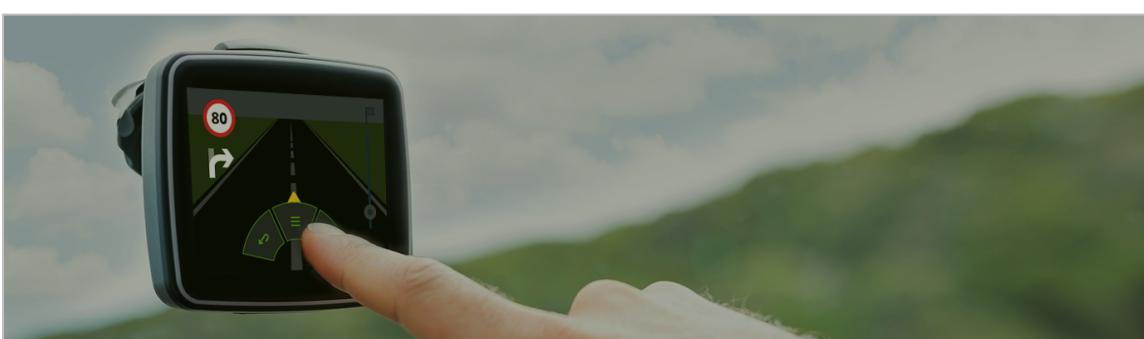
کیوت کوئیک (Qt Quick) یک فناوری مدرن رابط کاربری به سبک اعلانی است، که اجازه می‌دهند تحت زبان QML امکان نوشتن و طراحی رابط کاربری را به سبک خارق العاده و بسیار سریعی در اختیار کاربر قرار دهد و آن را با زبان C++ قدرتمند ترکیب و در نهایت در کمترین زمان ممکن بهترین طراحی را خلق نماید.

برنامه‌های تحت فناوری Qt Quick دارای یک بکاند ثابت از C++ را دارند که حتی اجازه دسترسی به تمامی امکانات دستگاهها را تحت آن فراهم می‌سازد. همچنین تحت این فناوری و ترکیب آن با قدرتمند ترین زبان این شرایط را ایجاد می‌کند که توسعه‌دهنده بتواند به پایین‌ترین لایه‌های ممکن تحت رابطه‌های برنامه‌نویسی دسترسی پیدا کند.

### معرفی زبان کیوام‌آل (QML) ویرایش ۲

یک زبان بر پایه جاوا اسکریپت (JavaScript) و برنامه‌نویسی اعلانی برای برنامه‌های کاربردی است. این زبان بخشی از فناوری Qt Quick (کیت UI توسعه یافته به وسیله شرکت DIGIA) است. کیو-ام-آل به طور عمده برای نرم‌افزارهای موبایل استفاده می‌شود. کیو-ام-آل بر سه پایه استوار است.

کیوت به عنوان حامل، اشیاء (مستطیل، دایره، مثلث، عکس...) و رفتارها (حالات، انتقال‌ها، انیمیشن‌ها) این المان‌ها می‌توانند برای ساخت قطعات پیچیده از قطعات ساده مانند کلیدها و لغزنده‌ها به منظور استفاده در برنامه‌های کاربردی و قابل دسترسی از طریق اینترنت استفاده شود. این المان‌ها همچنین می‌توانند با استفاده درون برنامه‌ای از برنامه‌نویسی جاوا اسکریپت و همچنین فریم‌ورک کیوت بر پایه زبان برنامه‌نویسی C++ توسعه پیدا کند.



استفاده از قابلیت‌های HTML یکی از بهترین و جذاب‌ترین مواردی می‌تواند باشد که در برنامه‌نویسی Mobile و Desktop بسیار جذاب خواهد بود هم‌این قابلیت‌ها دست به دست هم می‌دهند تا نویسنده در کمترین زمان بدون نیاز به صرف زمان بیشتر طراحی رابط‌کاربری را با بهترین فناوری فراهم سازد.

بنابر این با داشتن دانش C++ و اطلاعات کافی در رابطه با زبان‌های رابط‌کاربری همچون "HTML, CSS, JavaScript, QML" و آشنایی با محیط Qt که یک نوع چهارچوب (فریم‌ورک) ویژه‌ای برای این زبان است منجر به یک خروجی خارق العاده‌ای می‌شود : قدرت، سرعت، کیفیت، ارتباط مستقیم با سخت افزار! و در کنار این محیط با کیفیت بالا و همچنین طراحی مدرن همه و همه در خروجی نهایی برنامه‌شما حس بسیار عالی را منتقل خواهد شد.

## آشنایی با (نحو) سبک - سینتکس (Syntax) زبان QML

ممکن است در نگاه اول اینگونه تصور کنید که با زبان جدیدی مواجه شده‌اید که قرار است جایگزین زبان C++ شود! چرا که هنگام ایجاد پروژه کیوت کوئیک (Qt Quick) در اولین نگاه با کدهای جدیدی با نام QML مواجه می‌شوید. این تصور کاملاً اشتباه است، چیزی که می‌بینید یک زبان مخصوص طراحی رابط‌کاربری است و به هیچ عنوان تحت هیچ شرایطی جایگزین زبان C++ نخواهد بود و برعکس قرار است کمک بسیار زیادی با حفظ زمان برای توسعه‌دهنده امکاناتی را فراهم سازد تا با قدرت تمام و کیفیت بسیار متفاوت به توسعه پروژه خود به زبان C++ بپردازید.

خوب‌بختانه این زبان بر پایه یکی از روان‌ترین و جذاب‌ترین نحوهای موجود در دنیای برنامه‌نویسی است و شباهت بسیار زیادی به سبک JSON دارد. سبک و سیاق این زبان به شیوهٔ جاوا‌اسکریپت (JavaScript) بوده و حتی فراتر از آن به شکل خارق العاده‌ای روان‌تر و جذاب‌تر شده است. وجود این سبک باعث می‌شود برنامه‌نویس در زمان کد نویسی بسیار سریعتر و وسیع‌تر به توسعهٔ بخش تجربه‌کاربری (UX) و رابط‌کاربری (UI) بپردازد و در نهایت با رعایت اصول مهندسی چند لایه محصول استاندارد با کیفیت عالی را خلق نماید.

لازم بذکر است جهت استفاده و لینک‌کردن این زبان در پروژه می‌بایست از دستور زیر در qmake استفاده شود، کافی است این خط را در داخل فایل .pro اضافه کنید.

**QT += qml**

برای استفاده و ادغام آن با C++ دستور زیر استفاده شود:

```
#include <QtQml>
```

همچنین جهت وارد کردن ماژول در داخل Qt Quick توسط دستور زیر در دسترس قرار می‌گیرد:

```
import QtQml 2.2
```

در یک فایل QML ممکن است یک یا چند فایل به عنوان فایلهای اضافی یا ماژول در بالاترین قسمت از فایل به صورت زیر اضافه گردد :

- یک نوع از کlassen یا فضای نام تحت C++ که از قبل ثبت شده می‌تواند به عنوان یک افزونه (Plugin)، کلاس، نوع و یا غیر.
- یک دایرکتوری نسبی که می‌تواند شامل انواع تعریف شده‌ای از QML باشد.
- یک فایل جاوا‌اسکریپت (JavaScript)

شکل کلی شیوه‌های وارد کننده به فایل QML به صورت زیر است:

- `import Namespace VersionMajor.VersionMinor`
- `import Namespace VersionMajor.VersionMinor as SingletonTypeIdentifier`
- `import "directory"`
- `import "file.js" as ScriptIdentifier`

مثال به صورت زیر:

- `import QtQuick 2.0`
- `import Engine.Genyleap 1.0 as Cell`
- `import "../privateComponents"`
- `import "somefile.js" as Script`

دستورات وارد کننده در QML به گونه‌ای است که به شما اجازه می‌دهند به راحتی لایه‌های مختلف را به رابط کاربری خود متصل کنید که متشکل هستند از هسته، فایل‌های رابط سوم، جاوااسکریپت (Js)، سی‌اس‌اس (Css)، اچ‌تی‌ام‌ال (HTML) و غیره...

بنابراین جهت وارد کردن فرم کلی دستور وارد کننده به صورت زیر است :

```
import <ModuleIdentifier> <Version.Number> [as <Qualifier>]
```

۱. کلمه کلیدی import دستور اصلی جهت وارد کردن است. بخش <ModuleIdentifier> شناسه‌ای است که از قبل مشخص گردیده، برای مثال نام کلاسی در C++ که ثبت شده است. این گزینه، انحصاری بوده و فضای نام را در خود حمل می‌کند.
۲. گزینه <Version.Number> نسخه‌ی فایل یا همان فضای نام را در خود نگه می‌دارد و توسط آن می‌توان بر اساس نسخه مورد نظر گزینه مربوطه را جهت کاربردهای مختلف وارد QML کرد. برای مثال ممکن است دو کلاس با یک نام وجود داشته باشد در این صورت با تفاوت نسخه آنها می‌توان مشخص کرد که کدام یک از آنها برای چه منظوری تعریف شده است.
۳. و در نهایت <Qualifier> اجازه می‌دهد تا یک شناسه (نام مستعار) به صورت اختیاری را برای فضای نام مورد نظر تخصیص دهید. در این صورت فضای نام به صورت مستعار در جاوااسکریپت ثبت می‌شود و می‌توان از آن استفاده کرد.

مثالی از وارد کننده Qt Quick به فایل QML :

```
import QtQuick 2.9
```

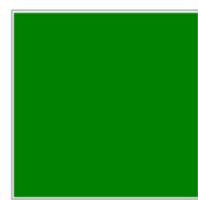
این امکان اجازه می‌دهد تا ماثول کیوت کوئیک (Qt Quick) را در داخل QML وارد کرده و از اعضای موجود در آن استفاده کنیم. برای مثال به کد زیر توجه کنید:

```

import QtQuick 2.0

Rectangle {
    width: 100
    height: 100
    color: "green"
}

```



در این مثال با استفاده از دستور وارد کننده مورد نظر دسترسی به کنترل شیء مستطیل (Rectangle) فراهم شده است که در نهایت با تعریف خصیصه‌های مرتبط با آن می‌توان شکل مورد نظر را ترسیم کرد.

مثال کلی از دستور **<Qualifier>** به صورت زیر خواهد بود:

```
import QtQuick 2.0 as Quick
```

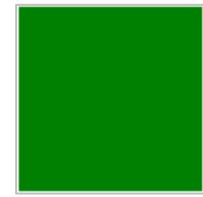
این قابلیت اجازه می‌دهد تا بر اساس فضای نام سفارشی شده (نام مستعار) توسط کاربر دسترسی به اشیاء موجود در ماژول Qt Quick فراهم شود که روشن استفاده به صورت زیر خواهد بود.

```

import QtQuick 2.0 as Quick

Quick.Rectangle {
    width: 100
    height: 100
    color: "green"
}

```



توجه داشته باشید که نام **Quick** در این کد یک نام مستعار به عنوان فضای نام بر اساس سلیقه کاربر است. این قابلیت زمانی مورد استفاده قرار می‌گیرد که یک یا چند ماژول هم نام باشند در این صورت با سفارشی سازی فضای نام می‌توان به هر دوی آنها دسترسی داشت. کافی است نام مستعار را نوشته و با کاراکتر نقطه (Dot) به زیر اعضا آن دسترسی پیدا کنید.

فرض کنید لازم است دو کلاس (دو فضای نام) با نام‌های مشابه وارد پروژه QML شود. در این صورت روش بالا بسیار کاربردی خواهد بود. همچنین زمانی که نیاز باشد از کنترل‌های مختلف ولی با یک فناوری مشابه استفاده کنید این روش باز هم کاربرد بسیار مناسبی خواهد داشت. به عنوان مثال در زیر سعی کرده‌ایم از دو فناوری مشابه برای ترسیم کنترل Button استفاده کنیم. این قابلیت تنها روشی است که می‌توان از طریق آن به هردوی آنها دسترسی داشته باشیم.

```

import QtQuick 2.9

import QtQuick.Controls 1.4 as Classic

import QtQuick.Controls 2.2 as Modern

```

```
Item {
```

```

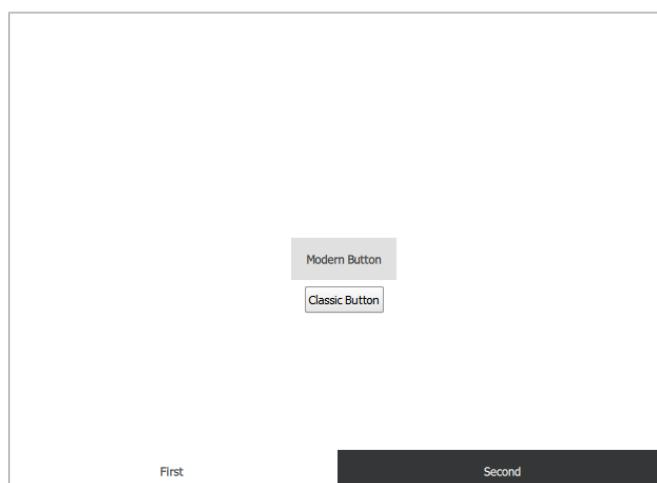
Classic.Button {
    x: 283
    y: 254
    text: 'Classic Button'
}

Modern.Button {
    x: 270
    y: 208
    text: 'Modern Button'
}

```



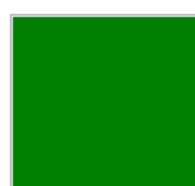
در کد بالا با انتخاب دو فناوری مشابه اما با سبک مختلف کلاسیک و مدرن آنها را توسط نام مستعار از هم جدا ساخته‌ایم تا دسترسی به آنها ممکن باشد. کافی است نام مستعار را فراخوانی و کنترل Button را انتخاب نماییم. نتیجه به صورت زیر نمایان خواهد شد که فرم ما دارای یک کنترل دکمه از نوع کلاسیک و یک نوع دیگری از نوع مدرن در در دسترس است.



## روش اعلام یا اظهار یک شیء در QML

اعلام یا اظهار و یا همان اصطلاح روتین بین برنامه‌نویس‌ها (Declarations) بسیار ساده است، کافی است بعد از وارد کردن مازول مورد مرتبط عضو یا شیء و یا کنترل مورد نظر را فراخوانی کنید. با نوشتن نام آن که حرف اول نیز بزرگ خواهد بود به آن و زیر خاصیت‌های شیء دسترسی خواهیم داشت. برای مثال کد زیر یک مستطیل با شناسه rec با عرض ۱۰۰ و ارتفاع ۱۰۰ با رنگ سیز ترسیم خواهد کرد.

```
Rectangle {
```



```

width: 100
height: 100
color: "green"
}

```

همچنین قابلیت نوشتمن در یک خط میسر است که در این روش می‌بایست خصوصیات مرتبط با شیء مورد نظر با سیمی‌کالون ";" جدا شوند.

```
Rectangle { width: 100; height: 100; color: "green" }
```

## اشیاء فرزند (Child Object)

هر یک از اشیاء می‌توانند با روش تو در توی اعلام کننده اشیاء فرزند یا اعضای درونی خود را بپذیرند. به این ترتیب که اعلام هر یک از اشیاء به شکل یک درخت می‌توانند دارای هر تعداد از اشیاء فرزند باشند.

برای مثال یک شیء مستطیل (Rectangle) می‌تواند از شیء فرزند با نام gradient را فراخوانی کند. مثال زیر روش فراخوانی و استفاده از اشیاء فرزند را معرفی می‌کند:

```

Rectangle {
    width: 100
    height: 100

    gradient: Gradient {
        GradientStop { position: 0.0; color: "#777777" }
        GradientStop { position: 1.0; color: "#CCCCCC" }
    }
}

```



همانطور که مشاهده می‌کنید با درج شیء Rectangle و فراخوانی خاصیت gradient آن می‌توان به شیء فرزند دسترسی و با اختصاص ویژگی‌های سفارشی نتیجه آن را مشاهده کرد.

این ویژگی تقریباً در تمامی اشیاء و کنترل‌های QML وجود دارد. جهت درک دقیق این مطلب مثالی از بکارگیری شیء فرزند در یک کنترل دکمه (Button) می‌زنیم که نیاز است ویژگی پس زمینه کنترل با یک رنگ متفاوت و سبک متن آن متناسب با آن نمایش داده شود.

```

import QtQuick 2.9

import QtQuick.Controls 2.2

Button {
    id: control

```

```

text: qsTr("C++")

contentItem: Text {
    text: control.text
    font: control.font
    opacity: enabled ? 1.0 : 0.3
    color: control.down ? "#ccc" : "#fff"
    horizontalAlignment: Text.AlignHCenter
    verticalAlignment: Text.AlignVCenter
    elide: Text.ElideRight
}

background: Rectangle {
    implicitWidth: 100
    implicitHeight: 40
    opacity: enabled ? 1 : 0.3
    border.color: control.down ? "#00589b" : "#0060aa"
    color: '#0060aa'
    border.width: 1
    radius: 2
}
}

```

C++

در این مثال با بکارگیری شیء دکمه (Button) دو شیء فرزند آن با نامهای contentItem برای بکارگیری و اعمال سفارشی سازی در متن دکمه و background در سبک و طرح کلیه دکمه را فراخوانی کرده‌ایم. توجه داشته باشید که برای استفاده از اعضای فرزند باید ابتدای نام آنها را با حرف کوچک آغاز کنید. سپس با جدا سازی آن توسط کarakتر دو نقطه ":" می‌توانید شیء مورد نظر خود را جهت اختصاص دادن به عضو فرزند تعیین کنید.

## سبک و روش اعمال نظر (Comment) در QML

همانطور که می‌دانید شیوه اعمال نظر در کنار هریک از کدها در هر زبان متفاوت است. بنابراین در زبان QML این سبک همانند JavaScript و C++ اعمال می‌شود.

- نظرات در یک خط با استفاده از کarakتر `//` آغاز شده و تا آخر همان خط ادامه می‌یابد.

- نظرات در چند خط با استفاده از کarakتر `*` آغاز شده و در آخر با استفاده از کarakتر `/` ادامه می‌یابد.

Text {

```

text: "Hello world!" //This is a basic greeting for C++/QML users.

/*
    This is an example comment of
    multi line mode.

*/
font.family: "Tahoma"
font.pointSize: 16
}

```

نظرات هنگام پردازش کد توسط موتور مربوطه در نظر گرفته نمی‌شوند. اعمال نظر در بین کدها همانطور که می‌دانید بسیار مفید است مخصوصاً زمانی که قرار است کدهای شما توسط توسعه‌دهنده دیگری مورد بررسی و توسعه قرار بگیرد.  
همچنین از این روش برای غیرفعال‌سازی بخشی از کد می‌توان استفاده کرد.

```

Rectangle {
    color: '#f23'
    width: 100
    height: 100
    //border.color: '#cccccc'
}

```

## صفت‌های اشیاء در QML

هر نوعی که در QML تعریف می‌شود دارای مجموعه‌ای از صفات‌هایی است که از قبل تعریف شده‌اند. البته لازم بذکر است توسعه‌دهنده، خود در صورت طراحی کامپوننت (جزئی) یا مازول مورد نظر می‌تواند اقدام به تعریف ویژگی‌های سفارشی نماید.  
مجموعه‌ای از نوع اشیاء (object-type) که با انواعی از ویژگی‌ها است به صورت زیر هستند:

- صفت شناسه (id)
- صفت اموال (اعضا)
- صفت سیگنال (Signal)
- صفت کنترل کننده سیگنال (Signal)
- تعریف یک روش خاص از صفت (property)

در ادامه به جزئیات ویژگی‌های فوق اشاره شده است.

## صفت شناسه (id)

هر یک از انواع اشیاء در QML دقیقاً یک صفت شناسه (id) را دارد. همچنین این صفت خودش توسط زبان برنامه‌نویسی فراهم می‌شود و نمی‌تواند دوباره برای شیء دیگری به صورت مشابه تعریف و یا باز نویسی شود.

برای مثال اگر قرار باشد دو شیء مستطیل (Rectangle) ترسیم شود می‌بایست شناسه این دو از هم متفاوت باشد. در غیر اینصورت با خطأ مواجه خواهد شد. لذا شناسه کاملاً انحصاری است.

در زیر مثالی از نحوه بکارگیری صفت شناسه (id) آورده شده است که در آن با تعریف شناسه rec1 و rec2 امکان شناسایی دو شیء مستطیل را فراهم ساخته‌ایم تا توسط رویداد کلیک در کنترل Button اعمال رنگ بر آن‌ها صورت بگیرد.

```
Rectangle {  
    id:rec1  
    width: 45  
    height: 45  
}  
  
Rectangle {  
    id:rec2  
    width: 45  
    height: 45  
}  
  
Button {  
    text: 'Set Color'  
    onClicked: {  
        rec1.color = "green"  
        rec2.color = "blue"  
    }  
}
```



نکته: تنها در صورتی می‌توان اشیاء موجود در سند QML را فراخوانی کرد که دارای شناسه باشند. در غیر اینصورت دسترسی به آن‌ها تنها در دامنه خودشان امکان پذیر خواهد بود.

## صفت خاصیت - ویژگی (property)

یک خاصیت (property) صفتی است از شیء‌ای که می‌تواند به یک مقدار ثابت یا یک عبارت پویا نسبت داده شود. به طور کلی آن می‌تواند توسط یک شیء دیگر اصلاح گردد، مگر اینکه یک نوع خاص QML به صراحت مجاز به آن نباشد.

یک خاصیت (property) شاید برای یک نوع C++ تعریف شده باشد که ثبت آن توسط کلمه‌کلیدی Q\_PROPERTY از کلاسی است که با سیستم نوع QML ثبت گردیده است. روش دیگری هم وجود دارد، یک خاصیت (property) سفارشی شده از یک شیء که شاید به عنوان یک شیء تعریف شده در سند QML باشد که به روش زیر خواهد بود:

```
[default] property <propertyType> <propertyName>
```

در این روش تعریف کردن یک شیء ممکن است مقدار خاصی را به اشیاء بیرونی درز دهد یا حفظ برخی از حالت‌های داخلی توسط آن راحت‌تر باشد. توجه کنید که نام خاصیت (property) باید فقط با کاراکتر کوچک آغاز شود و به جز کاراکتر متن، شماره و زیر خطی چیز دیگری را نمی‌پذیرد. همچنین باید به یک نکته ریز توجه داشته باشید که جاوااسکریپت (JavaScript) واژه‌هایی را در خود رزرو کرده است که استفاده از آن‌ها به عنوان نام سفارشی در یک خاصیت یا همان property پیشنهاد نمی‌شود.

در زیر برخی از این واژه‌ها (کلمات کلیدی) ذکر شده‌اند:

break	case	catch	enum
class	const	continue	implements
debugger	default	delete	interface
do	else	export	let
extends	finally	for	package
function	if	import	private
in	instanceof	new	protected
return	super	switch	public
this	throw	try	static
typeof	var	void	await
while	with	yield	abstract
boolean	byte	char	double
final	float	goto	int
long	native	short	synchronized
throws	transient	volatile	

برای استفاده از این ویژگی مثال زیر نمونه‌ای بسیار ساده و قابل درک است. برای مثال زمانی که نیاز باشد یک کامپوننت (جزئی) جداگانه در یک سند QML طراحی کنید با این روش می‌توانید با تعریف خاصیت (property) به کنترل یا اشیاء آن‌ها را سفارشی سازی نمایید. به مثال زیر توجه کنید:

```
import QtQuick 2.9
import QtQuick.Controls 2.2
Text {
    property string title
    property color textColor
    onTitleChanged: console.log("The title = " + title.toString())
    text : title; //Call text string from another input using by my new title property.
    color: textColor; //Call text color from another input using by my new textColor property.
}
```

در کد بالا با تخصیص دادن دو خاصیت (property) با نام `title` از نوع `string` (برای دریافت متن) و خاصیت دوم با نام `textColor` از نوع `color` (رنگ) جهت تخصیص کد رنگ مورد نظر اضافه کرده‌ایم. همچنین یک رویداد (`Event`) اختصاصی برای این گزینه با عنوان `onTitleChanged` تعريف کرده‌ایم که نتیجه نمایش آن محتوای ورودی به عنوان خواهد بود. سپس با دریافت مقادیر کلمات کلیدی خود را که `title` و `textColor` هستند به ترتیب برابر خاصیت `text` و `color` شیء `Text` از نوع QML می‌کنیم.

سپس برای استفاده از این خاصیت در صورتی که سندی با نام `MyNewFile.qml` داشته باشیم کافی است آن را در فایل مورد نظر فراخوانی کنیم. برای مثال در داخل فایل `main.qml` کد زیر را فراخوانی خواهیم کرد، برای این کار بر روی پروژه راست کلیک کرده و گزینه `Add New` را بزنید و سند مربوط به نوع QML را انتخاب و نام گذاری کنید. برای فراخوانی آن کافی است نام فایل را در سند مورد نظر وارد کنید. مانند مثال زیر:

```
MyNewFile {
    title : 'My name is Kambiz!'
    textColor : 'red'
}
```

با فراخوانی فایل `MyNewFile` به کنترل ساخته شده و سفارشی شده خود دسترسی خواهیم داشت که با تخصیص دادن مقادیر به دو خاصیت `text` و `textColor` (property) با نام‌های `text` و `color` مقادیر به سند اصلی انتقال می‌یابد.

نکته: نام فایل یا سندی که از نوع QML در صورتی که نیاز است به عنوان یک سند خارج از فایل وارد یک سند دیگری از QML شود می‌بایست با **کاراکتر بزرگ مشخص شود**.

## تعريف انواع معتبر توسط صفت ویژگی (property)

به طور کلی تمامی انواع تعریف شده پایه در QML می‌توانند به صورت سفارشی مورد استفاده قرار گیرند. برای مثال خاصیت‌های زیر انواعی معتبر هستند و که قالب تعریف به صورت زیر است:

```
property type name: value  
  
Item {  
  
    property int myNumber  
  
    property string myString  
  
    property url myUrl  
  
}
```

کُد بالا نمونه‌ای از ۳ نوع سفارشی از نوع‌ها int با نام جدید myNumber، نوع رشته با نام جدید myString و نوع url با نام جدید myUrl تعریف شده‌اند که تحت جدول زیر می‌توانید از انواع بیشتری در سفارشی سازی متغیرهای خود استفاده کنید:

نوع باینری با مقدار false یا true	bool
عدد با یک ممیز اعشاری، ذخیره شده به عنوان double	double
ارزش نام گذارده شده شمارشی	enumeration
نوع صحیح	int
لیستی از اشیاء QML	list
عدد با ممیز اعشار	real
رشته از نوع متن	string
نوع منبع یاب - مسیر یاب	url
نوع ویژه عمومی	var

ماژول QML ممکن است تحت زبان QML با انواع بیشتری توسعه یابد. برای مثال انواع پایه در ماژول Qt Quick به صورت زیر هستند که می‌توان از آن‌ها استفاده کرد:

پشتیبانی از نوع رنگ ARGB در QML	color
مقدار زمان	date
نوع فونت، با قابلیت تعریف فونت	font
ماتریکس ۴ در ۴	matrix4x4
خاصیت با مقادیر محورهای x و y	point

یک نوع چهار گانه (چهار سو) در محورهای scalar، x، y و z	<b>quaternion</b>
خاصیت با مقادیر محورهای x، y و ارتفاع	<b>rect</b>
مقدار اندازه با خصیصه‌های عرض و ارتفاع	<b>size</b>
یک نوع وکتور دو بعدی که دارای خاصیت محورهای x و y است	<b>vector2d</b>
یک نوع وکتور سه بعدی که دارای خاصیت محورهای x و y و z است	<b>vector3d</b>
یک نوع وکتور چهار بعدی که دارای خاصیت محورهای x و y و z و w است	<b>vector4d</b>

## صفت خاصیت - سیگنال (Signal)

یک سیگنال وظیفه اطلاع رسانی یک رویداد رخداده شده از طرف یک شیء را بر عهده دارد. برای مثال وقتی یک خصیصه‌ای تغییر می‌کند ممکن است یک اینیمیشن شروع به حرکت کرده و یا متوقف شود. و یا زمانی که یک تصویر به طور کامل دانلود (دربافت) شده باشد می‌توان توسط سیگنال از وضعیت آن مطلع شد. در نظر بگیرید نوع MouseArea یک رویدادی با نام clicked را دارد و زمانی که کاربر در محدوده آن کلیک کند سیگنالی را از خود ارسال خواهد کرد.

```
Item {
    width: 100; height: 100
    Text {
        id:messageOutPut
        width: 200
        height: 35
    }
    MouseArea {
        anchors.fill: parent
        onClicked: {
            console.log("Clicked!")
            messageOutPut.text = "Clicked!";
        }
    }
}
```

در کد بالا هنگامی که در محدوده کنترل MouseArea عمل کلیک شدن توسط کاربر صورت بگیرد سیگنال آن توسط onClicked ارسال می‌شود که در آن زمان می‌توان عملی را به عنوان چاپ یا تغییر و نمایش متن را انجام داد.

## تعريف ویژگی‌های سیگنال (Signal)

یک سیگنال ممکن است برای یک نوع شیء در C++ تعریف شود که توسط کلمه‌کلیدی مخصوص خود `Q_SIGNAL` از کلاسی که برای QML ثبت شده است باشد. البته روش دومی هم وجود دارد، یعنی یک سیگنال سفارشی برای یک نوع شیء که ممکن است توسط یک شیء تعریف شده در یک سند QML مشخص گردد و شکل کلی آن به صورت زیر خواهد بود:

```
signal <signalName>[([<type> <parameter name>[, ...]])]
```

در صورت تلاش برای اعلان دو سیگنال یا دو روش با یک نام مشابه در یک بلوک با خطاب مواجه خواهید شد. هرچند که ممکن است سیگنال تعریف شده جدید مجدداً مورد استفاده قرار بگیرد که در این صورت با احتیاط انجام شود چون ممکن است همان سیگنال با یک نام مخفی و از قبل تعریف شده موجود باشد.

تعريف سیگنال‌های سفارشی به عنوان رویدادها به صورت زیر است:

```
import QtQuick 2.9
import QtQuick.Controls 2.2

Text {
    //My new properties.

    property string title
    property color textColor
    onTitleChanged: console.log("The title = " + title.toString())
    signal clicked //new signal
    signal hovered()//new signal
    signal result(int x, int y)//new signal
    text: title; //Call text string from another input using by my new title
    property.

    color: textColor;//Call text color from another input using by my new
    textColor property.

    MouseArea {
        anchors.fill: parent
        hoverEnabled: true
        onHoveredChanged: hovered()
        onClicked: clicked
    }
}
```

```
    }  
}  
  
}
```

در این مثال سه نوع سیگنال با نام‌های `clicked`, `hovered` و `result` تعریف شده است. که در کد اصلی می‌توان از آن‌ها به صورت زیر استفاده کرد و بر اساس رویداد مربوطه کد مورد نظر را توسعه داد.

```
MyNewFile {  
  
    anchors.centerIn: parent  
    title : 'My name is Kambiz!'  
    textColor : 'red';  
  
    onHovered: {  
        console.log("Hovered!");  
    }  
}
```

### تعريف یک روش خاص از صفت (property)

یک روش می‌تواند به عنوان یک نوع C++ با وظیفه تابعی از یک کلاس تعریف شود. که برای QML توسط کلمه‌کلیدی `Q_INVOKABLE` ثبت می‌شود. و در واقع به عنوان یک شیار (Slot) از یک کلاس ثبت می‌شود. شکل کلی این روش برای سفارشی سازی آن به صورت زیر است:

```
function <functionName>([<parameterName>, ...]) { <body> }
```

این نوع می‌تواند به یک نوع QML اضافه شود و به صورت تنها از آن استفاده شود. در واقع روشی برای تعریف تابع در QML محسوب می‌شود که بیشترین کاربرد را خواهد داشت.

در مثال زیر نحوه تعریف یک تابع در QML در ساده‌ترین حالت ممکن آورده شده است :

```
function myFunction (){  
    return 'My name is Kambiz!';  
}
```

حال فرض کنید قرار است با کلیک شدن بر روی یک کنترل دکمه (Button) تابعی از طرف C++ و یا JavaScript صدا زده شود تا عمل چاپ و یا هر پردازش دیگری را انجام خواهد داد.

```
function myFunction (){  
    return 'My name is Kambiz!';  
}
```

```

Button {
    text: 'Print'
    onClicked: console.log(myFunction());
}

}

```

در این مثال تابعی از نوع جاوااسکریپت با نام myFunction تعریف شده است که با کلیک بر روی دکمه بازگشت نتیجه در کنسول چاپ خواهد شد. در نظر داشته باشید نیاز باشد تابعی از یک کلاس C++ را در این بخش صدا بزنید! وجود کلمه کلیدی قبل از نام تابع در C++ نیاز خواهد بود تا آن به عنوان یک شکاف (Slot) در نظر گرفته شود. که اطلاعات بیشتر در این زمینه در فصل‌های بعدی ارائه خواهد شد.

### پیوست (Binding) کردن صفات به یکدیگر

یک خاصیت می‌تواند به عنوان یک خصیصه ثابت از یک شیء تعریف و در اشیاء دیگر از همان استفاده شود. برای اینکار در صورتی که شیء ای زیر مجموعه شیء دیگر است می‌توان با گرفتن صفت از طریق والد (parent) به آن دسترسی پیدا کرد در غیر اینصورت می‌توان با استفاده از شناسه (id) شیء مورد نظر به آن دسترسی پیدا کرد.

برای مثال نمونه کد زیر نشان می‌دهد که دسترسی به خصوصیات یکی والد از طریق فرزند آن ممکن است:

```

import QtQuick 2.9

Rectangle {
    width: 100; height: 100
    Rectangle {
        width: 100
        height: parent.height
        color: "green"
    }
}

```

با در نظر گرفتن انتساب یک مقدار به ارتفاع والد بر فرزند با درج کلمه parent در کد می‌توان به خصیصهای آن دست یافت. همچنین در روش زیر می‌توان با استفاده از صفت شناسه (id) به این ویژگی استفاده کرد.

```

import QtQuick 2.9

Rectangle {
    id:rec1
    width: 100; height: 100
    Rectangle {
        id:rec2

```

```

width: 100
height: rec1.height
color: "green"
}

}

```

همانطور که مشخص است دسترسی به صفت با فراخوانی شناسهٔ شیء مورد نظر امکان پذیر است. این روش زمانی به کار می‌رود که اشیاء به صورت والد و فرزند زیر مجموعه هم و به روش درختی قرار نگرفته باشند. در این صورت این روش طبق نمونه کد زیر کاربرد بسیار زیادی خواهد داشت.

```

import QtQuick 2.9

Item {
    anchors.centerIn: parent
    Rectangle {
        id:rec1
        width: 100; height: 100
    }
    Rectangle {
        id:rec2
        width: 100
        height: rec1.height
        color: "green"
    }
}

```

## پشتیبانی از جاوااسکریپت (JavaScript) و ترکیب آن با QML

زبان QML با استفاده از سبک JSON اجازه می‌دهد تا عبارت‌ها و روش‌های مختلفی در جاوااسکریپت تعریف شوند، همچنین این قابلیت اجازه می‌دهد تا کاربر فایل‌های جاوااسکریپت (.js) را به عنوان یک تابع وارد کرده و از آن‌ها استفاده کنند. این قابلیت به طراحان و توسعه‌دهندگان قدرت نفوذ بسیار زیادی را اعطا می‌کند که توسط آن می‌توانند در هر دوی لایه‌های رابط‌کاربری و منطق نرم‌افزار تغییرات چشمگیری را اعمال کنند.

## عبارت جاوااسکریپت (JavaScript Expressions)

زبان QML قابلیت ادغام بسیار عمیقی را با جاوااسکریپت دارد، و اجازه کنترل سیگنال و روش‌های تعریف آن‌ها را در جاوااسکریپت فراهم می‌کند. یکی دیگر از ویژگی‌های هسته QML این است که توانایی برای مشخص کردن، تحقق بخشیدن و تقویت روابط بین خاصیت‌های اشیاء را با استفاده از سیستم اتصال‌دهنده صفات (property bindings) فراهم می‌کند، که هر کدام از آن‌ها می‌توانند با استفاده از جاوااسکریپت تعریف شوند.

## روش استفاده از جاوااسکریپت در سند QML

در ادامه توضیحاتی در رابطه با نحوه استفاده از جاوااسکریپت در کنار QML آورده شده است. محیط میزبانی جاوااسکریپت توسط QML فراهم می‌شود که می‌تواند ساختارهای صحیح (معتبر) و استاندارد جاوااسکریپت را اجرا کند که عبارتند از: عملگرهای شرطی، آرایه‌ها، تنظیمات متغیرها، حلقه‌های تکرار و ...

علاوه بر این خواص استاندارد دیگری از جاوااسکریپت به عنوان اشیاء عمومی (QML Global Object) موجود است که شامل تعدادی از روش‌های مفید و کمک کننده و ساده شده از رابطه‌های کاربری (UIs) است که با محیط QML ترکیب می‌شوند.

البته باید توجه کنید که محیط جاوااسکریپت ای که توسط QML فراهم می‌شود دشوار‌تر از محیط یک مرورگر اینترنتی است. برای مثال در QML شما نمی‌توانید اعضای عمومی جاوااسکریپت را تغییر و یا اضافه کنید. به طور منظم در جاوااسکریپت، ممکن است برای انجام آن به طور اتفاقی با استفاده از یک متغیر بدون تعریف آن انجام شود.

در QML این به عنوان یک استثناء تلقی خواهد شد، بنابراین تمامی متغیرهای محلی (local variables) می‌بایست به صراحت اعلان شوند. در ادامه مثالی از قوانین و محدودیت‌های جاوااسکریپت مشاهده خواهید کرد که در آن نحوه صحیح و صریح اعلام شدن متغیر را تحت جاوااسکریپت در QML مشاهده خواهید کرد.

## قوانين و محدودیت‌های محیط جاوااسکریپت (JavaScript)

در زیر مثالی آورده شده است که نشان می‌دهد در صورت رعایت نکردن نحوه اعلام متغیرهای محلی با خطأ مواجه خواهیم شد. بنابراین روش صحیح آن است که حین تعریف متغیر محلی آن را همراه با نوع آن تعریف کنیم.

```
x = 90; // Illegal modification of undeclared variable  
for (var y = 1; y < 10; ++y)  
x = x * y;  
console.log("Result: " + x);
```

در مثال فوق متغیر x که مقدار ۹۰ به آن انتساب داده شده است؛ یک متغیر محلی محسوب می‌شود که به صورت غیر نادرست تعریف شده است. بنابراین روش صحیح برای تعریف متغیر اینگونه است که نوع متغیر به همراه نام آن باید.

```
var x = 90;
```

```

for (var y = 1; y < 10; ++y)
x = x * y;
console.log("Result: " + x);

```

با در نظر گرفتن قوانین محدود کننده، می‌توان به ترکیب و استفاده عبارت‌های جاوااسکریپتی در QML پرداخت که در زیر مثالی از این روش آورده شده است که مشخص می‌کند ترکیب این دو باهم چگونه است:

```

import QtQuick 2.9

Rectangle {
    id: myButton
    width: 100; height: 100;
    color: mousearea.pressed ? "red" : "green"
    MouseArea {
        id: mousearea
        anchors.fill: parent
    }
}

```

در کد فوق ترکیب انواع QML و تخصیص عبارت‌های شرطی جاوااسکریپتی می‌توان بخشی از آن را مدیریت کرد. به عنوان مثال توسط جاوااسکریپت شرط تغییر رنگ شیء مستطیل (Rectangle) را با توجه به رویداد فشرده شدن محدوده ماوس اعمال شده است.

در واقع، هر عبارتی از جاوااسکریپت که (مهم نیست چقدر پیچیده باشد) ممکن است به عنوان یک صفت - پراپرتبی (property) (تعریف شود، تا زمانی به عنوان یک نتیجه از عبارت است که مقدار نوع آن بتواند به یک صفت اختصاص داده شود. این خاصیت می‌تواند در معرض تاثیرات جانبی قرار بگیرد. هرچند اتصال به شیوه پیچیده می‌تواند بر روی عملکرد آن‌ها تأثیر بگذارد زیرا می‌تواند خوانایی و کارآیی کد را کاهش دهد.

دو روش برای تعریف و تعیین صفت (property) وجود دارد: اولین و رایج‌ترین آن‌ها این است که به عنوان نمونه کد اول توضیح داده شد. اما روش مقدار دهی دوم به ندرت استفاده می‌شود که توسط یک تابع **Qt.binding** فراهم و مدیریت می‌شود. به مثال زیر توجه کنید:

```

import QtQuick 2.9

Rectangle {
    id: myButton
    width: 100; height: 100;

```

```

color: "green"

MouseArea {
    id: mousearea
    anchors.fill: parent
}

Component.onCompleted: {
    color = Qt.binding(function() {
        return mousearea.pressed ? "red" : "blue"
    });
}

```

در این روش که از آن به ندرت یاد می‌شود با استفاده از تابع از پیش تعریف شده `Qt.binding` می‌توان تابع مورد نظر جاوااسکریپتی را پیاده سازی و تعریف کرد.

## مدیریت سیگنال‌ها تحت جاوااسکریپت (JavaScript) در QML

همانطور که قبلاً هم توضیح داده شد در QML استفاده از سیگنال‌ها جهت مدیریت رویدادها ممکن است. همچنین مدیریت آنها در جاوااسکریپت فراهم می‌شود که مثال آن به شکل زیر خواهد بود:

```

import QtQuick 2.9

Rectangle {
    id: myButton
    width: 100; height: 100;
    color: "green"
    Text {
        id:myText
        text: 'Click Me!'
        anchors.centerIn: parent
    }
}

```

```

}

MouseArea {
    id: mousearea
    anchors.fill: parent
    onClicked: {
        myText.text = "C++ with Qt Quick" // کد جاوااسکریپت جهت
    }
}

```

## وارد کردن توابع جاوااسکریپتی تحت فایل (JavaScript) در QML

در QML می‌توان به راحتی فایل‌های جاوااسکریپتی را با پسوند (.js) وارد سند QML کرد. در این روش می‌توان با در نظر داشتن تابع مورد نظر در جاوااسکریپت آن را وارد مازول QML کرده و مورد استفاده قرار داد.

در مثال زیر یک فایل با نام myJSFile ایجاد کرده و کد زیر را در نظر بگیرید:

```

function sum( x, y ) {
    return x + y;
}

```

کد مربوطه تابعی است که عمل جمع دو پارامتر ورودی را انجام و آن را بازگشت می‌دهد. حال برای استفاده از یک تابع جاوااسکریپتی که در فایل با پسوند (.js) وجود دارد می‌بایست از روش وارد کننده (import) استفاده شود که به صورت زیر کد مربوط به آن آورده شده است:

```

import QtQuick 2.9
import "myJSFile.js" as JavaScript_Sum_Function

```

```

Rectangle {
    id: myButton
    width: 100; height: 100;
    color: "green"
    Text {
        id:myText
        text: 'Click Me!'
        anchors.centerIn: parent
    }
}

```

```

}

MouseArea {
    id: mousearea
    anchors.fill: parent
    onClicked: {
        myText.text = JavaScript_Sum_Function.sum(10,10);
    }
}

```

همانطور که مشاهده می‌کنید ابتدا فایل جاوااسکریپت را به روش استاندارد آن همراه با یک نام مستعار (**JavaScript\_Sum\_Function**) وارد سند QML کرده‌ایم.

سپس با فراخوانی آن دسترسی به تابع `sum` صورت گرفته و با وارد کردن پارامترهای مورد نظر نتیجهٔ نهایی را در اختیار خواهیم داشت. پارامترها می‌توانند شامل هر مقداری از کنترل‌ها باشند. برای مثال نام کنترل و صفت رشته می‌تواند به عنوان پارامتر ارسالی به تابع جاوااسکریپتی مورد استفاده قرار گیرد.

### نحوه اجرای یک کد جاوااسکریپتی در هنگام شروع یا راهاندازی اپلیکیشن یا یک کنترل خاص

فرض کنید نیاز باشد یک کد جاوااسکریپتی که می‌تواند یک تابع یا متغیر باشد تا در زمان اجرای برنامه مورد استفاده قرار دهد و به طور خودکار تابع مربوط به جاوااسکریپت شما اجرا شود. برای این منظور شیوهٔ بسیار کاربردی وجود دارد که تقریباً شامل حال همهٔ کنترل‌های موجود در ماژول `Qt Quick` است.

یک شیء QML سیگنالی را زمان کامل شدن فرآیند اجرا به رویداد (`Component.completed`) انتشار می‌کند. کد جاوااسکریپت بعد از پاسخدهی توسط رویداد `Component.onCompleted` مدیریت و اجرا می‌شود. در واقع بهترین مکان برای نوشتن کد مورد نظر جهت اجرا در حین اجرای برنامه یا بخشی از کد استفاده از قابلیت `Component.onCompleted` است که در بالاترین سطح یک شیء وجود دارد. زیرا این شیء هنگامی عمل می‌کند که محیط QML به طور کامل بارگذاری و ارتباط بین آن‌ها برقرار شده باشد.

به عنوان مثال کد زیر نمونه بسیار ساده‌ای از نحوه استفاده از این قابلیت است:

```

import QtQuick 2.9

Rectangle {
    function startupFunction() {
        // ... your startup code
    }
}

```

```

Component.onCompleted: startupFunction();

}

```

هر یک از اشیاء، کنترل‌ها و غیره... در QML شامل این رویداد هستند به عنوان مثال فرض کنید قرار است موقع اجرا و نمایش یک پنجره کدی اجرا شود که به صورت زیر می‌توان آن را پیاده سازی کرد.

```

import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.0

```

```

ApplicationWindow {

    id:mainWindow
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    Component.onCompleted: myFunction();

    function myFunction (){
        mainWindow.title = 'My name is Kambiz!';
    }
}

```

با توجه به کد فوق زمانی که پنجره اصلی برنامه، تحت فناوری Qt Quick اجرا شود تابعی با نام myFunction از نوع جاوااسکریپت اجرا خواهد شد که وظیفه آن همانطور که مشخص شده است تغییر عنوان پنجره اصلی برنامه خواهد بود.

## روش‌های ترکیب C++ و استفاده از آن در سند QML

زبان کیوام‌ال (QML) به گونه‌ای طراحی شده است تا به راحتی از طریق کدهای C++ توسعه یابد. کلاس‌های موجود در مازول Qt QML اشیاء کیوام‌ال را قادر می‌سازد تا با دخالت از طرف C++ بارگذاری شوند و این قابلیت در C++ ماهیت یکپارچه سازی موتور کیوام‌ال را با سیستم فرا اشیاء مرتبط با Qt فراهم می‌سازد تا به راحتی توابع و کلاس‌های سی‌پلاس‌پلاس به خوبی در کیوام‌ال مورد استفاده قرار بگیرند.

مازول QML چهار چوبی را برای اجرای برنامه‌های کیوام‌ال را فراهم می‌کند. در این میان این چهارچوب امکان این را فراهم می‌کند که کدهای کیوام‌ال حاوی کدهای جاوااسکریپت بوده و به راحتی با کدهای C++ ارتباط برقرار کند. مازول Qt QML کلاسی‌های سی‌پلاس‌پلاس را برای معرفی در چهار چوب کیوام‌ال ارائه می‌کند. کاربران می‌توانند از این کلاس‌ها استفاده کنند و همچنین می‌توانند کلاس‌های C++ را برای لایه‌های بالاتر یعنی کیوام‌ال ثبت کرده و از آنها جهت پردازش‌های نرم‌افزاری استفاده کنند.

به طور معمول برنامه تحت کیوام‌ال، در نقطه ورود با C++ ترکیب شده و توسط مازول QQmlEngine و استفاده از QQmlComponent اجازه‌این را می‌دهد که فایل کیوام‌ال بارگذاری شود.

به طور پیش فرض موتور Qt مازول QQmlContext را در بالاترین سطح ارزیابی قرار می دهد تا توابع و عبارت های مربوطه در سند کیوام ال تعريف شوند. ممکن است کاربر بخواهد یک تغییر یا گزینه جدیدی را در QQmlContext که توسط موتور فراهم شده است را اعمال کند. در این صورت باید آن را به صورت QQmlEngine::rootContext() درخواست کند تا به آن دسترسی داشته باشد.

کلاس QQmlEngine موتوری را فراهم می کند که می تواند مدیریت و اعمال سلسله مراتبی از مدیریت اشیاء را در سند کیوام ال انجام دهد. این گزینه به صورت یک ریشه از کیوام ال، به عنوان یک عبارت ارزیابی می شود و تضمین می کند که خواص اشیاء در زمان نیاز به درستی به روز رسانی خواهد شد. این ویژگی های منحصر بفرد اجازه می دهد تا توسعه یک برنامه با ترکیب و مخلوطی از QML، JavaScript و C++ با هماهنگی بسیار بالا صورت بگیرد.

## یکپارچه سازی QML و C++ انواع مختلفی از فرصت ها و توانایی ها را فراهم می سازد که به صورت زیر هستند:

- جداسازی پایه رابط کاربر (Front-End) از هسته و کدهای منطقی نرم افزار (Back-End) (با اجرای کیوام ال و JavaScript در محدوده اسناد کیوام ال با روش C++)
- استفاده و ارسال شماری از قابلیت های C++ برای QML به عنوان مثال ارسال بخشی از منطق برنامه، استفاده از یک مدل داده ای تعريف شده در سی پلاس پلاس، یا فراخوانی برخی از توابع و کتابخانه های نوع سوم در سی پلاس پلاس.
- قابلیت دسترسی در Qt Quick یا Qt QML تحت رابطه های برنامه نویسی C++ برای مثال، برای تولید تصاویر پویا استفاده از کلاس QtQuickImageProvider مورد استفاده قرار می گیرد.
- پیاده سازی انواع اشیاء QML اختصاصی کاربر برای استفاده در نرم افزارهای خاص، و یا برای توزیع اهداف دیگر.

توجه: برای فراهم کردن قابلیت استفاده از ویژگی های C++ در کیوام ال می بایست از کلاس مشتق شده اشیاء با کلمه کلیدی (QObject) استفاده شود تا ویژگی های ذکر شده فعال گردد. با توجه به یکپارچه سازی موتور QML با سیستم فرا اشیاء (meta objects)، خواص، روش ها و سیگنال های هر یک از کلاس های مشتق شده از طرف QML قابل دسترس هستند در واقع روشنی است که باعث می شود صفات های سی پلاس پلاس برای QML بیان شود.

## روش اول (در معرض قرار دادن و یا به اشتراک گذاری صفات سی پلاس پلاس در QML)

کیوام ال به راحتی می تواند با قابلیت های تعريف شده در سی پلاس پلاس توسعه پیدا کند. با توجه به ادغام موتور کیوام ال با سیستم ابر اشیاء در کیوت، هر یک از عملیات می تواند توسط روش کلاس مشتق شده برای QML در دسترس قرار بگیرند. این قابلیت امکان دسترسی انتقال و اشتراک گذاری داده های اجرایی سی پلاس پلاس را به طور مستقیم از طریق QML ممکن می سازد.

موتور کیوام ال این توانایی را دارد که از طریق سیستم (meta-object) توسط QObject شناسایی شود. این سیستم امکان برقراری ارتباط بین مکانیزم سیگنال و اسلات بین دو شیء و همچنین اطلاعات در زمان اجرا و سیستم و ویژگی های پویا (داینامیکی) را فراهم می سازد.

این بدان معنی است که هر یک از کدهای کیوام‌ال می‌توانند به شماری از اعضای کلاس‌های مشتق شده دسترسی پیدا کنند:

- خواص یا همان (Properties)

روش‌های (ارائه‌دهنده آن دسته از شکاف Slot)‌هایی که از نوع عمومی هستند و یا توسط کلمه‌کلیدی یا همان (ماکروی Q\_INVOKABLE) این روش برای توابع کاربرد خاصی دارد.

- سیگنال‌ها

علاوه بر این، انواع شمارشی نیز در دسترس هستند البته این در صورتی فراهم می‌شود که توسط کلمه‌کلیدی (ماکروی Q\_ENUMS) فراخوانی شوند.

## مالکیت و مدیریت بر نوع داده‌ها

هر داده‌ای که از طرف سی‌پلاس‌پلاس به کیوام‌ال منتقل می‌شود، چه به عنوان یک صفت، یک پارامتر و روش بازگشته یک مقدار، یا یک مقدار از پارامتر سیگنال، باید از نوعی باشد که موتور کیوام‌ال آن را پشتیبانی می‌کند. به طور پیشفرض، موتور مربوطه شماری از داده‌های سی‌پلاس‌پلاس را پشتیبانی می‌کند و زمانی که توسط QML مورد استفاده قرار می‌گیرند به صورت خودکار به مناسب‌ترین حالت تبدیل می‌شوند. علاوه بر این، کلاس‌های C++ که با نوع سیستمی QML ثبت شده‌اند می‌توانند به عنوان یک نوع داده مورد استفاده قرار گیرند.

## إفشاگری خواص (در معرض قرار دادن صفات (C++))

ویژگی یا صفت (property) را می‌توان برای هر یک از کلاس‌های ارث بری شده از QObject با استفاده از کلمه‌کلیدی یا ماکروی (Q\_PROPERTY) مشخص کرد. یک صفت (property) داده‌ای از عضو کلاسی است که با تابع خواندن (read) و نوشتمن (write) به صورت اختیاری تعیین می‌شود. بنابراین تمامی صفت‌های یک کلاس مشتق شده از طرف QML در دسترس هستند.

در ادامه مثالی آورده شده است که در نظر داریم در آن با تعریف یک کلاس و اعضای آن صفتی را با عنوان نام (name) در سی‌پلاس‌پلاس تعریف کنیم که در کیوام‌ال به آن دسترسی داشته باشیم.

در این مثال کلاسی با نام MyClass در سی‌پلاس‌پلاس تعریف خواهد شد که از کلاس QObject ارث بری کرده است. البته توجه نمایید که سرآیند مربوط به کلاس QObject را وارد کرده باشید.

```
class MyClass : public QObject // کلاس ارث بری شده از کیوآبجکت
```

```
{
```

```
public:
```

```
    MyClass();
```

```
    Q_OBJECT // ماکرو جهت فعال‌سازی ویژگی‌های مکانیزم متا‌آبجکت در کیوام
```

```

Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)

public:

    void setName(const QString &a) {
        if (a != m_name) {
            m_name = a;
            emit nameChanged();
        }
    }

    QString getName() const {
        return m_name;
    }

signals:

    void nameChanged();

private:

    QString m_name;
};

```

ماکروی ([Q\\_OBJECT](#)) باید در بخش خصوصی کلاس اعلان شود تا از مکانیزم سیگنال و اسلات در کیوت پشتیبانی کند. سپس با در نظر گرفتن کلمه‌کلیدی - ماکروی ([Q\\_PROPERTY](#)) صفت مورد نظر را که قرار است نام (name) باشد تعریف خواهیم کرد. توجه کنید که نوع [QString](#) نشانگر این است که این صفت از نوع رشته بوده و قرار است با خصیصه‌های READ جهت دریافت اطلاعات و WRITE برای نوشتمن بر روی آن مشخص شود که به ترتیب با نام‌های متغیر `setName` و `getName` تعریف شده‌اند. همچنین در آخر آن سیگنالی با نام `nameChanged` با دستور NOTIFY برای بررسی تغییر وضعیت مقدار صفت مشخص شده است.

در ادامه توسط تابع `void` با نام `setName` از نوع `QString` تعريف شده است که با دریافت مقدار متغیر `m_name` از نوع `QString` عملیات لازم راجه‌ت مشخص‌سازی مقدار و ارسال سیگنال `nameChanged` به عنوان یک اطلاع رسان از وضعیت تغییر مقدار قرار گرفته است.

تابع `void` از نوع `getName` که می‌تواند نوع بازگشتی را داشته باشد، به عنوان یک رشته مقدار `m_name` را بر می‌گرداند. توجه داشته باشید که تابع `void` از نوع `getName` که می‌تواند نوع بازگشتی را داشته باشد، به عنوان یک رشته مقدار `m_name` را بر می‌گرداند. توجه داشته باشید که تابع `void` از نوع `nameChanged` signal نشانگر آن است که این صفت به عنوان سیگنال تعريف شده است.

در نهایت برای فایل سرآیند `myclass.h` کد نهایی به صورت زیر خواهد بود:

```

#ifndef MYCLASS_H
#define MYCLASS_H
#include <QObject>

class MyClass : public QObject

```

```

{

public:
    MyClass();

Q_OBJECT

Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)
public:
    void setName(const QString &a);
    QString getName() const;

signals:
    void nameChanged();

private:
    QString m_name;

};

#endif // MYCLASS_H

```

همچنین کد مربوط به فایل سورس myclass.cpp به صورت زیر است:

```

#include "myclass.h"

MyClass::MyClass() {}

void MyClass::setName(const QString &a) {

    if (a != m_name) {
        m_name = a;
        emit nameChanged();
    }
}

QString MyClass::getName() const {

    return m_name;
}

```

حال اگر به صورت خلاصه کلاس را بررسی کنیم، می‌توان گفت که با ارسال یک مقدار از نوع رشته به عضو name در کلاس با نام MyClass می‌توانیم عملیات فوق را در QML انجام دهیم. تنها مواردی که در این حالت اضافه شده‌اند ماکرو‌های Q\_OBJECT و Q\_PROPERTY هستند که در کتابخانه STL سی‌پلاس‌پلاس وجود ندارند و این نشانگر این است که این دو همانند دیگر کلاس‌های سی‌پلاس‌پلاس در کیوت برای هماهنگی و ترکیب این زبان با QML فراهم شده است.

در این مرحله کلاس مورد نظر در سی‌پلاس‌پلاس آماده شده است و تنها لازم است آن را برای QML تعریف کنیم. که به شکل زیر در فایل main.cpp

کدهای مشخص شده را اعمال خواهیم کرد.

```

#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext> //include QQmlContext class

```

```

#include "myclass.h" //include myclass header file

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);

    QGuiApplication app(argc, argv);

    MyClass myclass; //Class decelare

    QQmlApplicationEngine engine;

    engine.rootContext()->setContextProperty("myClass", &myclass);

    engine.load(QUrl(QLatin1String("qrc:/main.qml")));

    return app.exec();
}

```

## مالکیت و مدیریت بر نوع داده‌ها

کلاس مورد نظر را با نام `myclass` وارد منبع اصلی کرده‌ایم، البته توجه داشته باشید وارد کردن کلاس `QQmlContext` همانطور که در صفحات قبلی توضیح داده شده است مهم است. لذا وارد کردن آن نباید فراموش شود.

در ادامه نام اصلی کلاس خود را `MyClass` نوشته و آن را اعلام می‌کنیم. اما باید توجه داشته باشیم توسط فراخوانی خصیصه `rootContext` از طرف موتور، صفت مورد نظر را که شامل کلاس `myclass` است تعریف کنیم که در کد زیر آمده است:

```
engine.rootContext()->setContextProperty("myClass", &myclass);
```

این کد با مراجعه به آدرس مربوط به کلاس مورد نظر و تعریف یک نام مستعار برای آن که قرار است در QML با همان نام فراخوانی شود را مشخص می‌کند.

```
; (نام کلاس اعلام شده & ، "نام مستعار")
```

**نکته مهم:** نام مستعار می‌بایست با حرف کوچک آغاز شود تا در QML با مشکل مواجه نشود، این قانون به خاطر ایجاد تداخل در صفت‌های از پیش تعریف شده QML است که در صفحات قبلی به آن‌ها اشاره کرده‌ایم.  
در ادامه کد کیوام‌آل به صورت زیر خواهد بود :

```

Text {
    width: 100; height: 100
    anchors.centerIn: parent
    text: myClass.name // invokes myClass::name() to get this value
    Component.onCompleted: {
        myClass.name = "Kambiz" // invokes myClass::setName()
    }
}

```

در کد مذکور نام مستعار کلاس سی‌پلاس‌پلاس، (myClass) ذکر شده و صفت name برابر صفت text در کنترل Text شده درج شده است که در ادامه آن با در نظر گرفتن رویداد بارگذاری شدن کنترل مقدار "Kambiz" را در آن اختصاص داده‌ایم که در نهایت با اجرای برنامه مقدار کنترل نمایان خواهد شد.

## روش دوم و روش پیشنهادی (تعريف کلاس‌های C++ و ثبت آن در QML)

همانطور که مشاهده می‌کنید در روش اول تمامی خاصیت‌های سی‌پلاس‌پلاس را می‌توان در معرض استفاده قرار داد، آن روش به تنها‌یی روش خوبی بشمار می‌رود اما روش دیگری نیز وجود دارد که علاوه بر قابلیت‌های روش اول این امکان را فراهم می‌کند تا کلاس‌ها و توابع سی‌پلاس‌پلاس را در موتور QML به گونه‌ای ثبت کنیم که از لحاظ انعطاف‌پذیری همانند خود QML در دسترس باشند.

برای ثبت کردن یک کلاس مشتق شده سی‌پلاس‌پلاس به عنوان شیء QML، باید از تابع qmlRegisterType() استفاده شود. با این روش می‌توان کلاس مورد نظر را همراه با فضای نام مورد نظر وارد سند QML کرد. پس از آن کاربر می‌تواند فضای نام مورد نظر را به منظور استفاده وارد نماید. ابتدا مثال قبلی را که در روش اول زده شده است با این روشی پیاده خواهیم کرد سپس جهت توضیحات بیشتر برای سفارشی‌سازی و دسترسی به تمامی قابلیت‌های ممکن مثال‌های دیگری خواهیم زد. کد برای فایل سرآیند MyClass.h به صورت زیر خواهد بود:

```
#ifndef MYCLASS_H
#define MYCLASS_H
#include <QObject>

class MyClass : public QObject
{
public:
    MyClass();
    Q_OBJECT

    Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)

public:
    void setName(const QString &a);
    QString getName() const;

signals:
    void nameChanged();

private:
    QString m_name;
};

#endif // MYCLASS_H
```

به خاطر اینکه قرار است ابتدا مثال قبلی را با روش رجیستر (ثبت) کردن تکرار کنیم هیچ تغییراتی در فایل‌های سرآیند و سورس سی‌پلاس‌پلاس کدهای قبلی نداده‌ایم و تنها قرار است در فایل اصلی یعنی main.cpp با مقداری تغییرات مواجه شویم.

همچنین کد مربوط به فایل سورس myclass.cpp به صورت زیر است:

```
#include "myclass.h"
```

```

MyClass::MyClass() {}

void MyClass::setName(const QString &a) {
    if (a != m_name) {
        m_name = a;
        emit nameChanged();
    }
}

QString MyClass::getName() const {
    return m_name;
}

```

حال برای ثبت کردن این کلاس در فایل QML می‌بایست در فایل main.cpp از دستور زیر استفاده کنیم؛ دقت کنید که نسخه‌نگاری برای مازول شما برای دسترسی در سند QML نیز مهم است :

```
qml.RegisterType<MyClass>("QML.Genyleap", 1, 0, "MyClass");
```

```

#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext> //include QQmlContext class
#include "myclass.h" //include myclass header file

int main(int argc, char *argv[])
{
    QCOREAPPLICATION::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);

    QQmlApplicationEngine engine;
    qml.RegisterType<MyClass>("QML.Genyleap", 1, 0, "MyClass");
    //Registration C++ class for QML

    engine.load(QUrl(QLatin1String("qrc:/main.qml")));
    return app.exec();
}

```

در کد مذکور با فراخوانی تابع **qml.RegisterType** ابتدا کلاس مورد نظر را صدا زده و سپس نام مناسبی که قرار است به عنوان یک مازول ورودی در QML مشخص شود را وارد می‌کنیم. سپس با استفاده از جدا کننده کاما (" ") پارامتر دوم که مربوط است به شماره اصلی نسخه مازول را مشخص و سپس شماره دوم - فرعی آن را وارد خواهیم کرد. در نهایت با جدا سازی و پارامترنهایی به عنوان نام مستعار که قرار است به عنوان یک نوع در QML شناخته شود را مشخص خواهیم کرد. برخلاف روش اول وارد کردن این گزینه حساسیتی در رابطه با کوچک یا بزرگ بودن حرف آن ندارد.

در نهایت برای استفاده از این کلاس ثبت شده از طرف سی‌پلاس‌پلاس باید به روش زیر در سند QML آن را وارد کنیم :

```

import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.0
import QML.Genyleap 1.0 // My custom module

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    MyClass { //C++ Class
        id: cppClass
    }

    Text {
        width: 100; height: 100
        anchors.centerIn: parent
        text: cppClass.name// invokes myClass::name() to get this value
        Component.onCompleted: {
            cppClass.name = "Kambiz" // invokes myClass::setName()
        }
    }
}

```

در کد مذکور که مرتبط با روش دوم است جهت دسترسی به کلاس می‌بایست بر خلاف روش اول ابتدا کلاس ثبت شده به عنوان مازول را وارد سند QML کنیم. به صورت زیر :

```
import Genyleap 1.0 // My custom module
```

همانطور که مشخص است همان چیزی که در فایل main.cpp تعریف شده با همان نام و همان نسخه تایید می‌شود. حال در ادامه به مهم‌ترین قسمت خواهیم رسید که ممکن است بعضی از کاربران بدون در نظر گرفتن آن اقدام به فراخوانی کلاس نمایند. در این صورت کلاس فراخوانی خواهد شد اما در نگاه اول متوجه آن نخواهید شد، سپس در مرحله اجرا با این خطا مواجه خواهید شد که این کلاس رجستر (ثبت) نشده است و یا وجود ندارد. بنابراین، دقت کنید که بعد از وارد کردن مازول در داخل کنترل مادر (اصلی) نام کلاس را به عنوان کنترل تعریف شده فراخوانی و شناسه‌ای را برای آن در نظر بگیرید:

```
MyClass { //C++ Class
    id: cppClass
}
```

حال در داخل مکان مناسب برای مثال رویداد کلیک، رویداد بارگذاری، تایмер یا هر آنچه که نیاز دارید نام کلاس را نوشته و تابع مورد نظر آن را انتخاب کنید تا بتوانید از آن کلاس در QML استفاده کنید.

نکته : این روش یک روش بسیار بی‌نقص و عالی است، بنابراین مثال بعدی در رابطه با نحوه تعریف تابع با استفاده از ماکروی **Q\_INVOKABLE** را توضیح خواهیم داد.

قبل از مثال لازم است توضیحی در رابطه با ماکروی **Q\_INVOKABLE** بدهیم. این ماکرو همانطور که از نامش مشخص است مخصوص فراخوانی تابع سیپلاس برای کیوام‌ال است. که از طریق سیستم فرا اشیاء اجازه تعریف تابع برای فراخوانی را می‌دهد. این ماکرو قبل از کلمه کیدی نوع تابع نوشته می‌شود. و شکل کلی آن به صورت زیر است:

```
[Q_INVOKABLE] [void] [yourFunctionName]
```

```
class MyClass : public QObject
{
    Q_OBJECT

public:
    MyClass();

    void normalMethod(); // روش عادی
    Q_INVOKABLE void invokableMethod(); // روش ویژه فراخوانی تابع
```

با در نظر گرفتن این روش برای اینکه در یک کلاس زیر مجموعه‌ای از آن که به عنوان یک تابع در نظر گرفته شده است را در کیوام‌ال فراخوانی و مورد استفاده قرار دهیم نیاز است تا از این قانون پیروی نماییم.

کد زیر نمونه مثالی است که قرار است به جزئیات این شیوه بپردازد:

```
class MyClass : public QObject
{
    public:
    MyClass();
    Q_OBJECT

    public:
        Q_INVOKABLE QString print();
}
```

کلاس فوق دارای تابعی است که قرار است متنی را از نوع رشته چاپ کند. بنابراین طبق قانون ذکر شده با استفاده از ماکروی فراخوانی کننده **Q\_INVOKABLE** آن را ثبت می‌کنیم. مراحل تکمیل سازی کلاس بر اساس قوانین سیپلاس پلاس در فایل سورس myclass.cpp صورت می‌گیرد که به صورت زیر خواهد بود:

```
#include "myclass.h"

MyClass::MyClass() {}

QString MyClass::print() {
    return "My name is Kambiz";
}
```

دقیق کنید که در فایل سورس نباید ماکروی `Q_INVOKABLE` مجدداً فراخوانی شود. در نهایت با استفاده از تابع `qmlRegisterType` روش ثبت کلاس را دنبال کرده و به تابع مورد نظر در QML دسترسی خواهیم داشت.

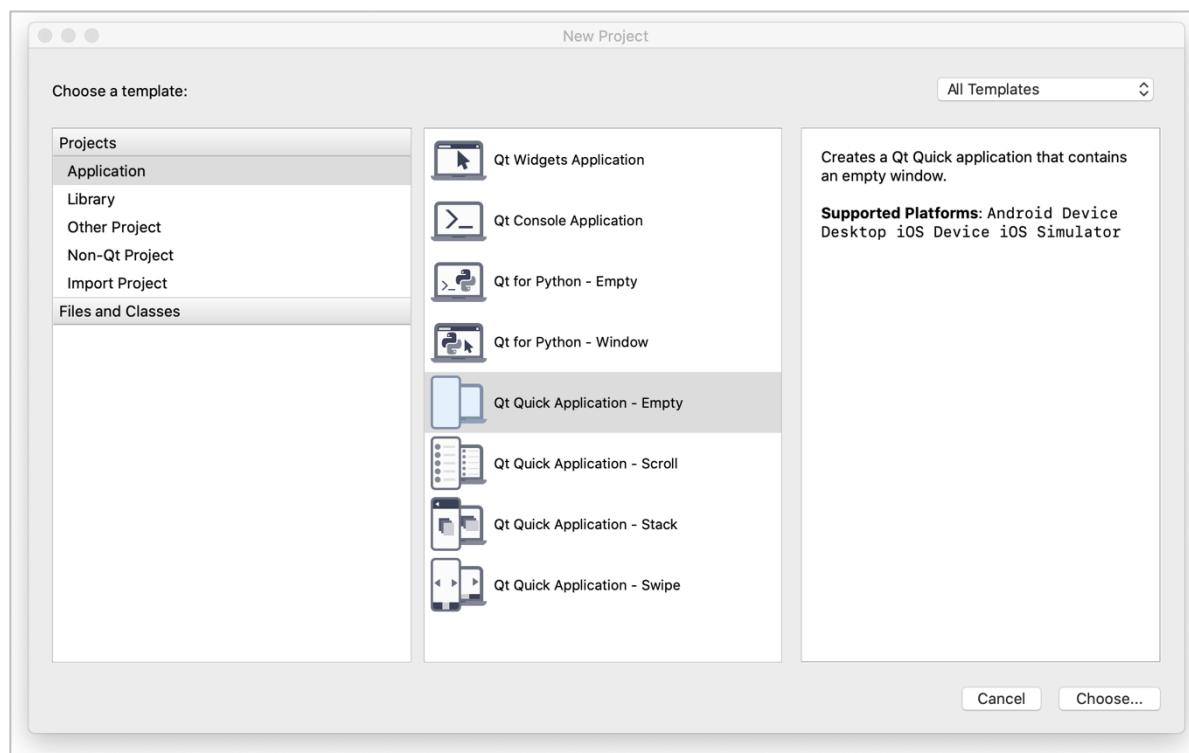
در نهایت کد مربوط به سند QML به صورت زیر خواهد بود:

```
Button {  
    text: "Print"  
    onClicked: console.log(cppClass.print());  
}
```

## فصل سوم

### معرفی انواع پروژه‌ها C++ تحت فناوری کیوت کوئیک (Qt Quick)

هرچند در کتاب مقدماتی به انواع پروژه‌های موجود در کیوت اشاره شده است، اما لازم است بار دیگر به گزینه‌های مرتبط با غیر فناوری Qt Quick اشاره کنیم تا تفاوت بین آنها با کیوت کوئیک را بهتر درک کنیم.



### پروژه از نوع Qt Widgets Application

این نوع پروژه اولین و ساده‌ترین نوع پروژه‌های مرتبط با محیط Qt است که با انتخاب این گزینه شما می‌توانید پروژه خود را بر پایه‌ی فناوری سنتی ویجت (Widget) طراحی کنید. پروژه‌های ایجاد شده در این نوع از فناوری به صورت سنتی بوده و توسعه‌دهنده، نسبت به نیاز خود می‌تواند از کنترل‌ها و ابزارهایی که به صورت پیشفرض بر روی آن قرار دارند استفاده کند. از مزایای این نوع از پروژه (سازگاری با این نوع پروژه‌ها در تمامی تجهیزات به بهترین صورت است و حتی در ضعیفترین سخت افزارها و تجهیزات ممکن برنامه شما به خوبی اجرا خواهد شد.)

از معایب آن می‌توان به عدم پشتیبانی از موتورهای تولید کننده تصاویر قدرتمند اشاره کرد که در نهایت طراحی در رابطه با UI/UX تحت این فناوری بسیار جذاب و مدرن نیست.

### پروژه از نوع Qt for Python - Empty

این نوع پروژه‌ها با توجه به نام پایتون، تنها با ایجاد یک فایل شامل یک فایل با محتوای پایه‌ای از کدهای کیوت برای زبان برنامه‌نویسی پایتون است.

### پروژه از نوع Qt for Python - Window

این نوع پروژه‌ها با توجه به نام پایتون، تنها با ایجاد یک فایل شامل یک فایل و پنجره اصلی با محتوای پایه‌ای برای ایجاد یک اپلیکیشن تحت کدهای کیوت برای زبان برنامه‌نویسی پایتون است.

### پروژه از نوع Qt Console Application

همانطور که از نامش مشخص است بیشتر در محیط کنسول و کدهای زیر برنامه‌ای استفاده می‌شود. این محیط بیشتر برای برنامه‌ای پایه و ساده که نیازی به محیط‌های گرافیکی ندارند توصیه می‌شود.

### پروژه از نوع Qt Quick Application - Empty

این نوع از پروژه یکی از مدرن‌ترین نوع پروژه‌های ممکن بر پایه فناوری Qt Quick است که علاوه بر پشتیبانی از زبان QML از زبان C++ جهت طراحی رابطه‌ای مدرن در بخش Front-End را پشتیبانی می‌کند و برای برنامه‌های تحت موبایل، Embedded ها و دسکتاپ مناسب است. مزیت این نوع پروژه‌این است که قابلیت‌ها و انعطاف‌پذیری‌های بسیار زیادی پشتیبانی می‌کند.

### پروژه از نوع Qt Quick Application - Scroll

این نوع از پروژه دقیقاً همان نوع قبلی است، با تفاوت اینکه نوع پیشفرض کنترلی آن بر پایه کنترل ScrollView تنظیم شده است.

### پروژه از نوع Qt Quick Application - Stack

این نوع از پروژه دقیقاً همان نوع قبلی است، با تفاوت اینکه نوع پیشفرض کنترلی آن بر پایه کنترل StackView تنظیم شده است.

### پروژه از نوع Qt Quick Application - Swipe

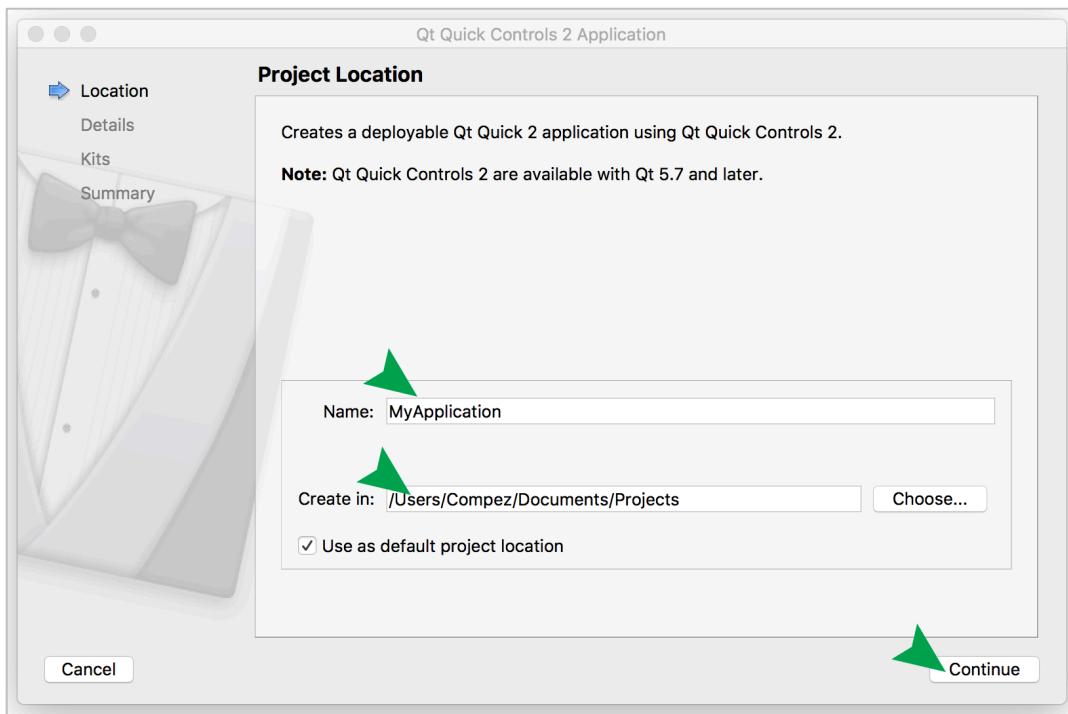
این نوع از پروژه دقیقاً همان نوع قبلی است، با تفاوت اینکه نوع پیشفرض کنترلی آن بر پایه کنترل SwipeView تنظیم شده است.

**نکته:** پروژه‌های تحت فناوری کیوت کوئیک، با پشتیبانی از موتورهای بومی سیستم‌عامل‌ها و سازگاری بسیار بالا با سخت‌افزارها و رابطه‌ای برنامه‌نویسی پیشرفته، بالاترین سرعت ممکن را همراه با طراحی‌های خلاقانه ارائه می‌کند. نرخ رندر در این نوع پروژه‌ها به ۶۰ فریم و حتی بیشتر هم می‌رسد که حتی در سیستم‌های ضعیف پاسخگو است.

## آغاز ایجاد پروژه تحت C++ و فناوری Qt Quick

برای شروع به منوی File و گزینه New File or Project مراجعه کنید. حال در این قسمت شما انواع پروژه‌ها و فایل‌هایی که می‌توانید توسط Qt ایجاد کنید با توجه به نسخه‌ای که نصب کرده‌اید فعال و قابل انتخاب است که ما در این آموزش از پروژه Empty - Qt Quick Application استفاده می‌کنیم، لذا مناسب‌ترین گزینه ممکن برای تولید نرم‌افزار است.

حال با ادامه این مرحله به صورت زیر نام پروژه و مسیری که مایل هستید پروژه در آن مکان ذخیره شود را انتخاب کنید:



توجه کنید، در صورتی که قرار است برای پروژه‌های تحت پایه لینوکس و یونیکس مانند اندروید و آی‌اواس، برنامه‌نویسی کنید حسابیست به کاراکترهای فاصله، کوچک و بزرگ بودن بسیار مهم است لذا دقت نکردن به این موارد در حین تولید برنامه بسیار مشکل ساز خواهد شد. بنابراین از مرحله اول به این موارد توجه کافی داشته باشید.

بعد از مشخص کردن مسیر پروژه نوع پوسته آن را تحت Qt Quick مشخص خواهیم کرد که دارای گزینه‌های Default به صورت پیش فرض و دو استایل Universal و Material است. در صورتی که پروژه خود را قصد دارید به صورت چند-سکویی توسعه دهید نگران این نباشد که آیا قرار است در پلتفرم دیگر پروژه را مجددا تنظیم نمایید. چرا که شما تحت ابزار مدیریتی Qbs و همچنین نوع پیشرفته آن با نام pro می‌توانید تمامی استناد یک پروژه را به خوبی بر اساس نوع هدف در پلتفرم‌های مورد نظر مدیریت نمایید.

نکته: در صورتی که قصد دارید برنامه خود را سفارشی سازی نمایید بهتر است این قسمت را بر روی گزینه Default قرار دهید و سپس Continue را انتخاب کنید.

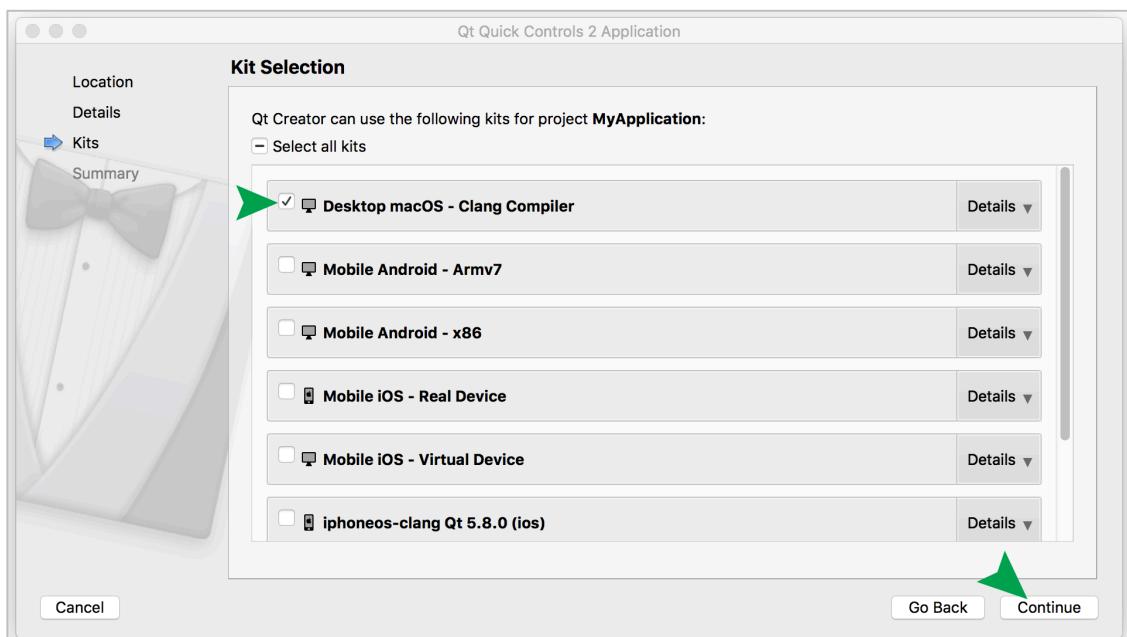
• همانطور که می‌دانید در کیوت امکان سفارشی سازی قالب پروژه در بخش UX و UI بسیار انعطاف پذیر است و تحت فناوری Qt Quick و ترکیب زبان‌های QML و JavaScript, CSS, HTML قادر به ایجاد انواع پوسته‌ها در پروژه خواهیم بود. بنابراین اگر شما هدفتان تولید برنامه‌ای برای محیط‌های کامپیوترهای رومیزی (دسکتاپ) و یا سفارشی سازی و ایجاد یک رابط کاربری منحصر بفرد است گزینه پیشفرض (Default) مناسب‌ترین گزینه خواهد بود.

• اگر هدف شما تولید و توسعه اپلیکیشن‌های موبایل و تبلت در پلتفرم اندروید است می‌توانید با بهره‌گیری از پوسته توصیه شده توسط Google را با نام Material استفاده قرار دهید.

• توجه داشته باشید در صورتی که برای iOS یا AppleWatch و دیگر محصولات اپل برنامه‌نویسی می‌کنید بهتر است همان پوسته پیش فرض را انتخاب کنید.

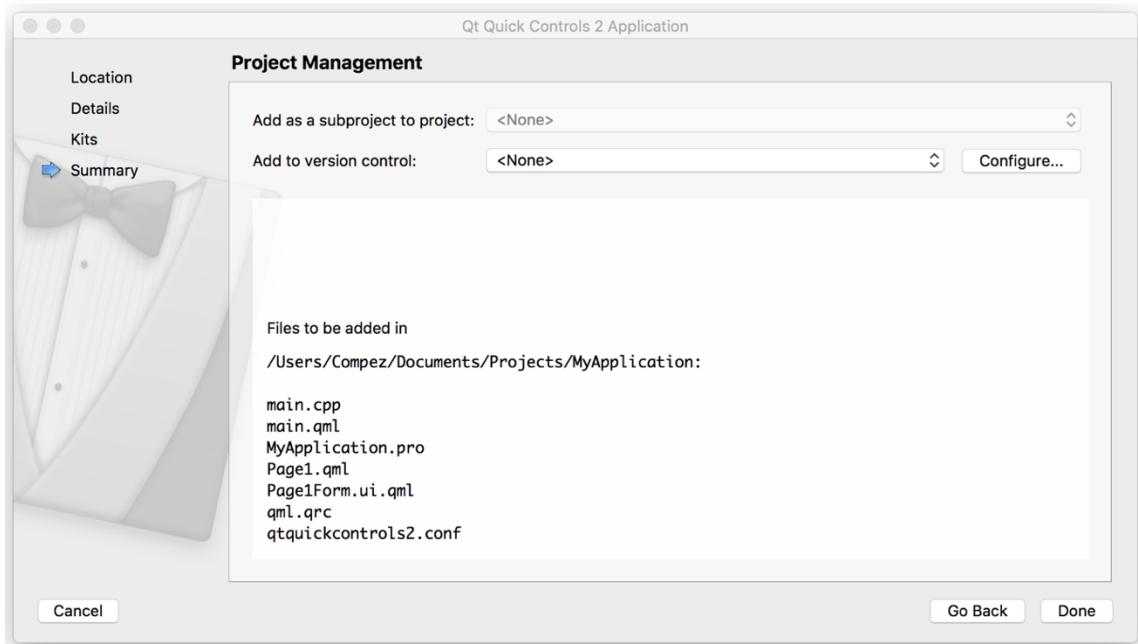
• اگر بر روی پلتفرم ویندوز ۱۰ و ویندوز فون محصولی ارائه خواهد داد بهتر است از پوسته Universal استفاده شود.

در این بخش در صورتی که کیت (Kit) های مربوطه بر روی سیستم به درستی نصب و شناسایی شده باشند با تصویر زیر مواجه خواهیم شد که بر اساس نوع پلتفرم و نسخه کیوت نصب شده قابل انتخاب هستند. هدف ما انتخاب نسخه دسکتاپ بر روی پلتفرم macOS است.

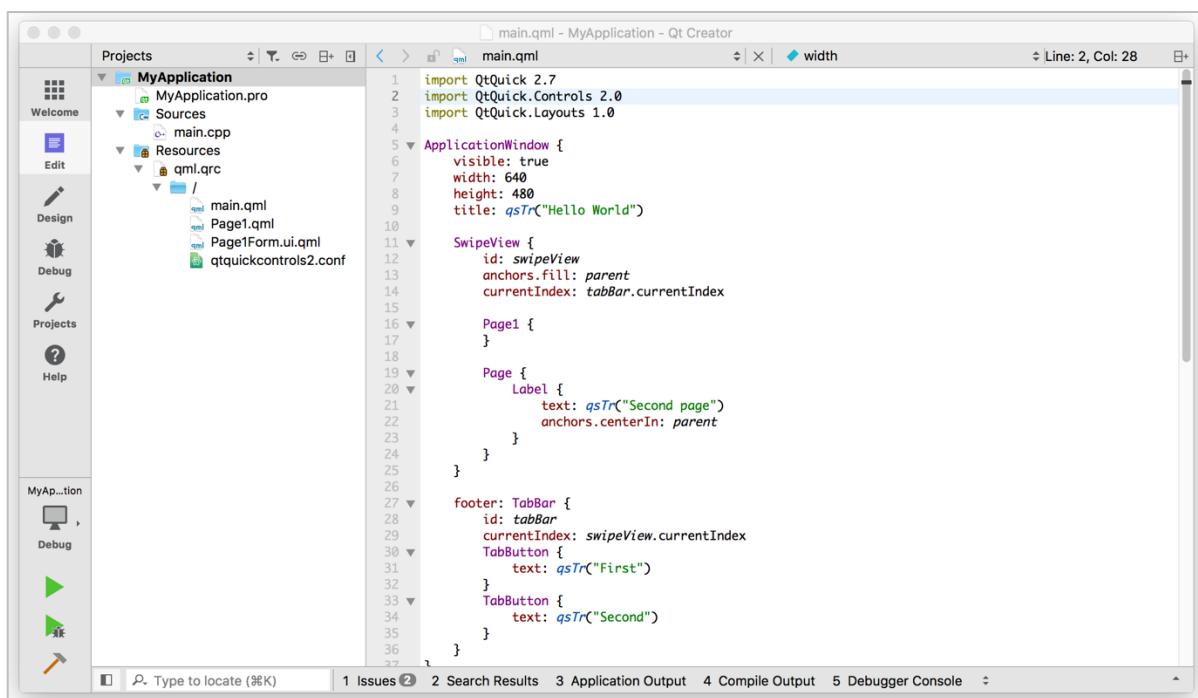


در مرحلهٔ بعد بخش مدیریت پروژه بر روی سرویس دهنده‌ها نمایان می‌شود.

برای مثال اگر در نظر دارید پروژه خود را بر روی میزبانی همچون GitHub ایجاد کنید می‌توانید با انتخاب آن در بخش "Add to version control" آن را مشخص کنید. در کتاب ما قصد داریم بر روی میزبان محلی کامپیوتر خودمان آن را ایجاد کنیم بنابراین تغییراتی اعمال نخواهیم کرد.



گرینهً مورد نظر را انتخاب و در مرحله بعد گزینه Done را انتخاب می‌کنیم تا وارد پروژه ایجاد شده به صورت زیر شویم.



توجه داشت باشید که پروژه به صورت پیش فرض دارای فایل main.cpp است. فایل main.qml اصلی پروژه بوده و فایل اصلی مربوط به QML است. دو فایل Page1.qml و Page1Form.ui.qml مربوط به کامپوننت‌های پیش‌فرض هستند که وجود آن‌ها زیاد ضروری نیست و تنها به صورت آزمایشی ایجاد شده‌اند. لذا جهت توسعه سفارشی ممکن است به آن‌ها نیازی نداشته باشیم.

در این بخش ما یک پروژه از نوع **Qt Quick Application** ایجاد کرده‌ایم. در پروژه‌های Qt ما چند نوع فایل پروژه‌ای در اختیار داریم که با پسوند .pri و .pro که هر دوی این فایل‌ها توسط Qt قابل شناسایی هستند و به عنوان فایل اصلی پروژه شما در نظر گرفته می‌شوند مشخص شده‌اند.

اگر روی فایل `MyApplication.pro` کلیک کنید با کدهای زیر مواجه خواهیم شد:

```
QT += qml quick (ماژول‌های موجود بر روی پروژه)
CONFIG += c++11 (مقدار مربوط به پشتیبانی از سی‌پلاس‌پلاس)
SOURCES += main.cpp (نگه دارنده فایل‌های سورس سی‌پلاس‌پلاس)
RESOURCES += qml.qrc (نگه دارنده سورس فایل‌های کیو‌ام‌ال و ...)
# Additional import path used to resolve QML modules in Qt Creator's code model
QML_IMPORT_PATH =  
  
#Default rules for deployment. (قوانين مرتبط با نوع خروجی و پیکربندی آن)
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

فراموش نکنید فایل `.pro` مهمترین قسمت پروژه است که در فراخوانی فایل‌ها و رفرنس‌ها کاربرد دارد برای مثال اگر بخواهیم از دیتابیس و دستورات SQL استفاده کنیم ابتدا باید ماژول آن در این فایل فراخوانی شود.

پوشه Headers وظیفه نگهداری تمام فایل‌های C++ از نوع `h`. یا همان `header` را بر عهده دارد. پوشه Sources وظیفه نگهداری تمام فایل‌های C++ از نوع `cpp`. یا همان `Source` را بر عهده دارد. پوشه Forms وظیفه نگهداری تمام فایل‌های مربوط به طراحی را دارد که پسوند فایل‌های طراحی در `mainwindow.ui`. هستند. به صورت `mainwindow.ui` که فایل طراحی پروژه شما به عنوان یک فرم در نظر گرفته شده است. همچنین پوشه `qml.qrc` وظیفه نگهداری فایل‌های QML را بر عهده دارد.

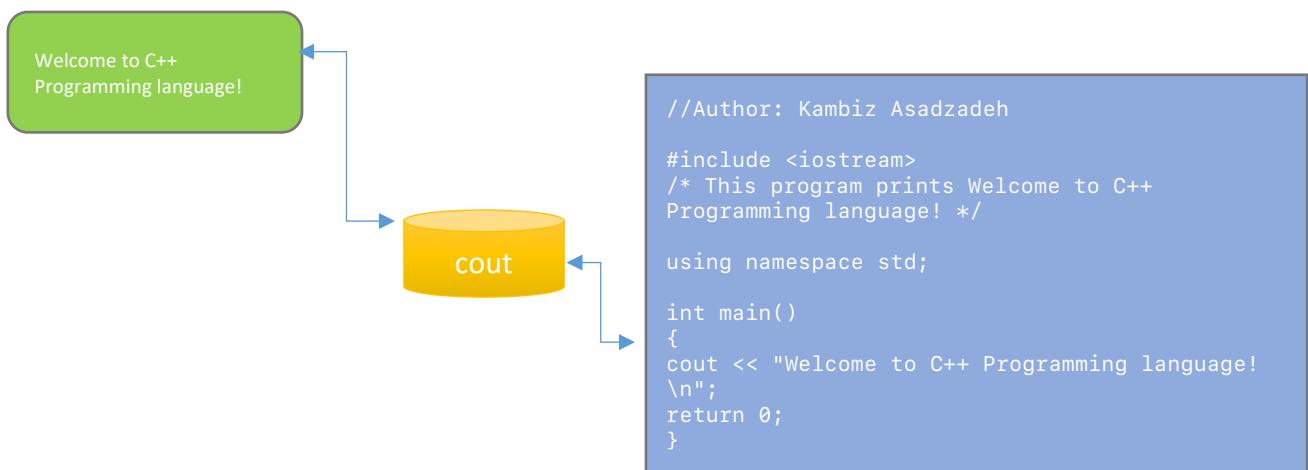
معمولًا برای اضافه کردن ماژول‌های مورد نظر به پروژه از فایل `.pro` از نماد `=+=` استفاده می‌کنیم. با این عملگر به کامپایلر اعلام می‌کنیم که این ماژول به پروژه اضافه شود. برای حذف ماژول از عملگر `=-=` استفاده می‌شود. به مثال زیر توجه کنید:

```
QT -= gui
QT += core network
```

با توجه به کتاب برنامه‌نویسی C++ که توضیحات در رابطه با قوانین و دیگر ویژگی‌ها و روش‌های برنامه‌نویسی در این زبان اشاره شده است ما در این آموزش‌ها با در نظر گرفتن آن کتاب و داشتن اطلاعات از قبل و کافی در این مورد برای آموزش این کتابخانه توسط C++ می‌پردازیم.

همانطور که می‌دانید در هر زبان برنامه‌نویسی می‌توان ساده‌ترین برنامه را در چند خط کوتاه ایجاد و نتیجه را نمایش داد. مراحل ابتدائی ساختار تولید برنامه در C++ به صورت زیر ساده‌ترین کدنویسی به زبان C++ در نسخه جدید را توضیح می‌دهد.

به شکل زیر توجه کنید:



کد بالا یک کد بسیار ساده از زبان سی‌پلاس‌پلاس مدرن، بدون دخالت کتابخانه کیوت است. در کد زیر که بعد از ایجاد پروژه از نوع Qt Quick در فایل main.cpp مشاهده خواهیم کرد.

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);

    QGuiApplication app(argc, argv);
    QQmlApplicationEngine engine;
    engine.load(QUrl(QLatin1String("qrc:/main.qml")));
    return app.exec();
}
```

## معرفی کلاس QGuiApplication

کلاس **QGuiApplication** شامل رویدادهای اصلی و حلقه‌ای است. به طور کلی هر جایی که رویدادی در بک‌اند برنامه رخ دهد، سیستم به صورت خودکار آن را از طریق این کلاس فراخوانی و مدیریت می‌کند. همچنین این کلاس تنظیمات وسیعی از پروژه را شامل می‌شود و تمامی آن‌ها را مدیریت می‌کند. باید توجه داشت، هر پروژه‌ای که شامل لایه‌های گرافیکی کاربر است چه شامل یک پنجره باشد و یا چندین عدد از آن‌ها هیچ تفاوتی نخواهد کرد. پروژه، دارای یک کلاس **QGuiApplication** خواهد بود که به تنهایی همه آن‌ها را مدیریت خواهد کرد. بنابراین برای تولید توسعه و مدیریت پروژه‌های تحت رابط کاربری گرافیکی این کلاس شرط اول است.

## معرفی کلاس QQmlApplicationEngine

کلاس **QQmlApplicationEngine** شبیه مناسبی را برای بارگذاری فایل‌های QML فراهم می‌کند. این کلاس به تنهایی وظيفة ترکیب کلاس **QQmlComponent** و **QQmlEngine** را بر عهده دارد و مناسب‌ترین روش بارگذاری تحت سیگنال بر فایل‌های QML را فراهم می‌سازد. همچنین این کلاس برعی از قابلیت‌های مرکزی نرم‌افزار را برای QML فراهم می‌سازد که یک برنامه ترکیبی C++/QML/QML فراهم می‌سازد. بنابراین جهت برقراری ارتباط بین C++ و فناوری Qt Quick وجود این کلاس نیز شرط است.

```
int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);
    QQmlApplicationEngine engine;
    engine.load(QUrl(QLatin1String("qrc:/main.qml")));
    return app.exec();
}
```

تابع `main` که تابع اصلی C++ بوده و اولین تابعی است که در برنامه اجرا خواهد شد.

## معرفی کلاس QCoreApplication

کلاس **QCoreApplication** حلقه‌ای از رویدادها را برای برنامه‌های تحت کیوت از نوع بدون رابط کاربری فراهم می‌سازد و می‌توان گفت مهمترین و اصلی‌ترین کلاسی است که برای ساخت یک برنامه تحت کیوت نیاز خواهد بود. در کد بالا این کلاس همراه با تعیین ویژگی پشتیبانی از مقیاس پذیری در کیفیت تصاویر با تراکمی بسیار زیاد (با وضوح بالا) را مشخص کرده است که شکل کلی تعریف ویژگی در پروژه تحت این کلاس به صورت زیر است:

```
void QCoreApplication::setAttribute (
    Qt::ApplicationAttribute attribute,
    bool on = true )
```

کد بعد از آن که شامل کلاس **QtGuiApplication** است، پروژه را برای شناسایی از نوع پروژه گرافیکی آماده سازی می‌کند.  
با توجه به توضیحاتی که در رابطه با **QQmlApplicationEngine** داده شد، نوع جدیدی از آن با نام مستعار **engine** ساخته شده است که مقدار دهی اولیه برای بارگذاری فایل QML و اجرای آن توسط C++ را فراهم می‌سازد و شکل کلی استفاده برای بارگذاری فایل QML به صورت زیر است:

```
void QQmlApplicationEngine::load(const QString &filePath)
```

توجه داشته باشید که در این بخش خاصیت **load** یک شکاف (اسلات) - Slot با دسترسی عمومی است. و وظیفه آن شناسایی ریشه و بارگذاری فایل QML بر اساس مسیری است که در قالب رشته برای آن ارسال می‌شود.

## معرفی تابع exec

در نهایت تابعی جهت مدیریت و اعلام وضعیت دستور خروج به برنامه با نام **exec** وجود دارد. این تابع از نوع **int** بوده و وظیفه‌اش این است که وارد حلقه‌ای در تابع **main** شود که منتظر فراخوانی تابع خروج (**exit**) باشد. پس از آن که مقدار خروجی با اجرای تابع **exit** مشخص شد ارزش آن برابر ۰ خواهد بود.

توضیحات مربوطه در رابطه با تابع اصلی **main** در C++ بود که در آن اشاره مستقیمی به فایل **main.qml** جهت بارگذاری اصلی ترین فایل QML شده است. کد این فایل به صورت زیر شامل جزئیات مخصوص در زبان QML است.

```
import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.0

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    SwipeView {
        id: swipeView
        anchors.fill: parent
        currentIndex: tabBar.currentIndex

        Page1 {
        }

        Page {
        }
    }
}
```

```

Label {
    text: qsTr("Second page")
    anchors.centerIn: parent
}
}

footer: TabBar {
    id: tabBar
    currentIndex: swipeView.currentIndex
    TabButton {
        text: qsTr("First")
    }
    TabButton {
        text: qsTr("Second")
    }
}
}

```

در کد موجود قوانین QML صدق می‌کند، ابتدا دستورات وارد کنندهٔ مازول‌هایی که در آن استفاده شده‌اند مشخص شده است. شکل کلی یک دستور جهت وارد کردن و شناساندن مازول و یا کامپوننت مورد نظر به صورت زیر است:

**[نام مستعار, توصیف کننده در صورت نیاز] شماره ویرایش-نسخه مازول <u\_نام مازول>**

برای درک بهتر به دستور زیر دقت کنید:

```
import QtQuick 2.9
```

در این دستور با رعایت قانون وارد کردن ابتدا کلمهٔ کلیدی import و سپس نام مازول و بعد از آن شماره مربوط به نسخه آن مشخص شده است. این دستور فراخوانی مازول Qt Quick را بر عهده دارد.

همچنین دستور بعدی به صورت زیر :

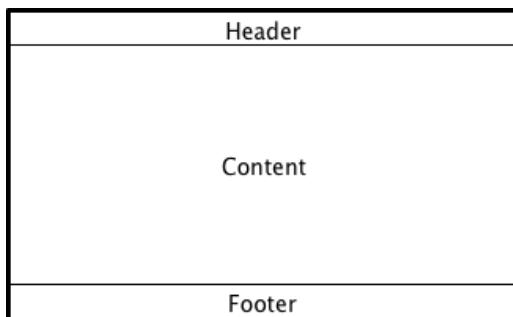
```
import QtQuick.Controls 2.2
```

وظیفهٔ فراخوانی مازول Qt Quick Controls از نسل جدید را فراهم می‌کند.

دستور بعدی طبق کد زیر :

```
import QtQuick.Layouts 1.0
```

وظیفه فراخوانی مازولی را بر عهده دارد که شامل تمامی کنترل‌های لایه است. گزینه **ApplicationWindow** یک نوع از QML است که وظیفه تولید پنجره اصلی مربوط به برنامه را بر عهده دارد و دارای ویژگی‌های اختصاصی است.



ساختار کلی این نوع به صورت زیر است:

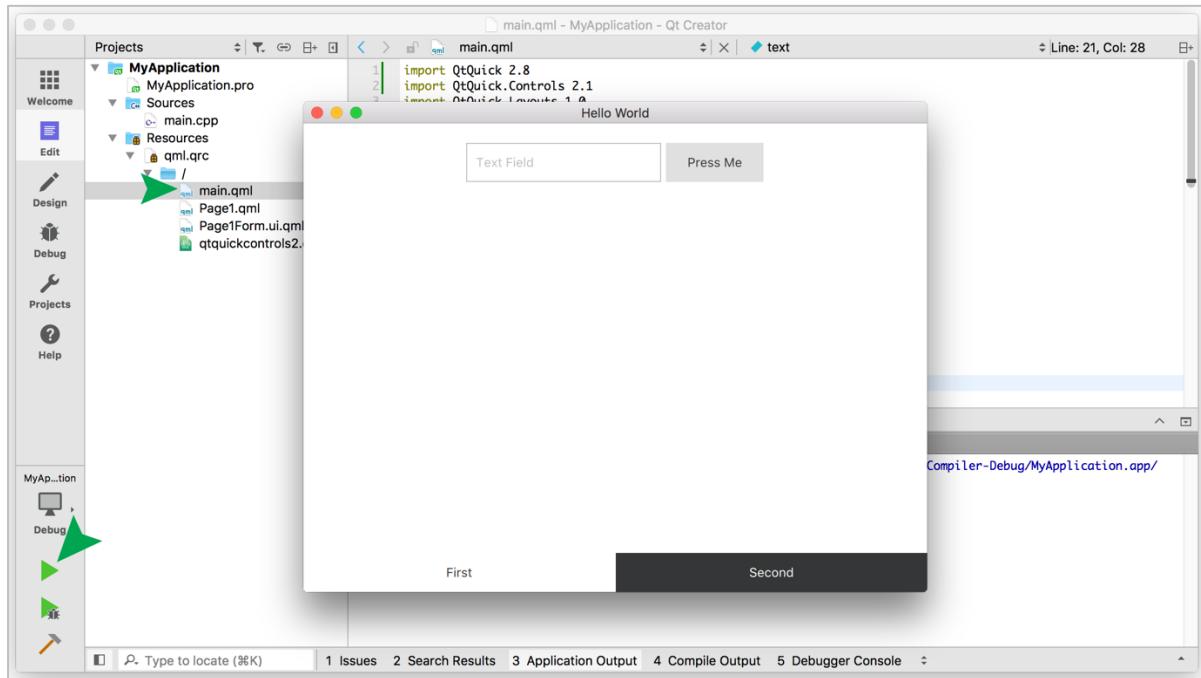
```
ApplicationWindow {  
    visible: true  
    header: ToolBar {  
        // ...  
    }  
    footer: TabBar {  
        // ...  
    }  
    StackView {  
        anchors.fill: parent  
    }  
}
```

import QtQuick.Controls 2.2	دستور وارد کننده مازول
Qt 5.8	قابل دسترس از نسخه
Window	ارت بری از

مشخصه‌های `visible`, `width`, `height` و `title` به صورت پیش‌فرض موجود است، که برای مشخص سازی اندازه‌های عرضی و ارتفاع برنامه با مقدار عددی `int` و همچنین مشخصه بعدی وظیفه نمایش دادن آن را بر عهده می‌گیرد که با مقدار `true` و `false` ارزش گزاری می‌شوند. همچنین مشخصه `title` با قابلیت پذیرش رشته عنوان برنامه شما را نمایش می‌دهد.

نکته: تمامی اعضای مرتبط در QML با کarakتر اول کوچک آغاز می‌شوند.

ویژگی header و footer دو بسیار مهم محسوب می‌شوند که برای استفاده و سفارشی سازی با این دو خاصیت از نسخه دوم کوئیک می‌تواند بهره‌مند شد و آن را با استفاده از انواع QML مانند tabBar ، ToolBar و حتی Item ، Rectangle و غیره... سفارشی سازی کرد. نهایتاً با کلیک بر روی دکمه اجرای پروژه، ابتدا کدهای مربوطه کامپایل و سپس در حالت اشکال‌زدایی اجرا و تصویر زیر نمایان خواهد شد که به جزئیات آن می‌پردازیم.



فراموش نکنید که این یک برنامه ساده است، حال برای اینکه برنامه خود را سفارشی سازی کنیم و آن را بر اساس سلیقه خود توسعه دهیم، می‌بایست با تمامی خاصیت‌ها و قوانین QML آشنا شده و آن‌ها را با روش‌های صحیح به کار بگیریم.

## انواع کنترل‌ها، منوها و دیگر اشیاء

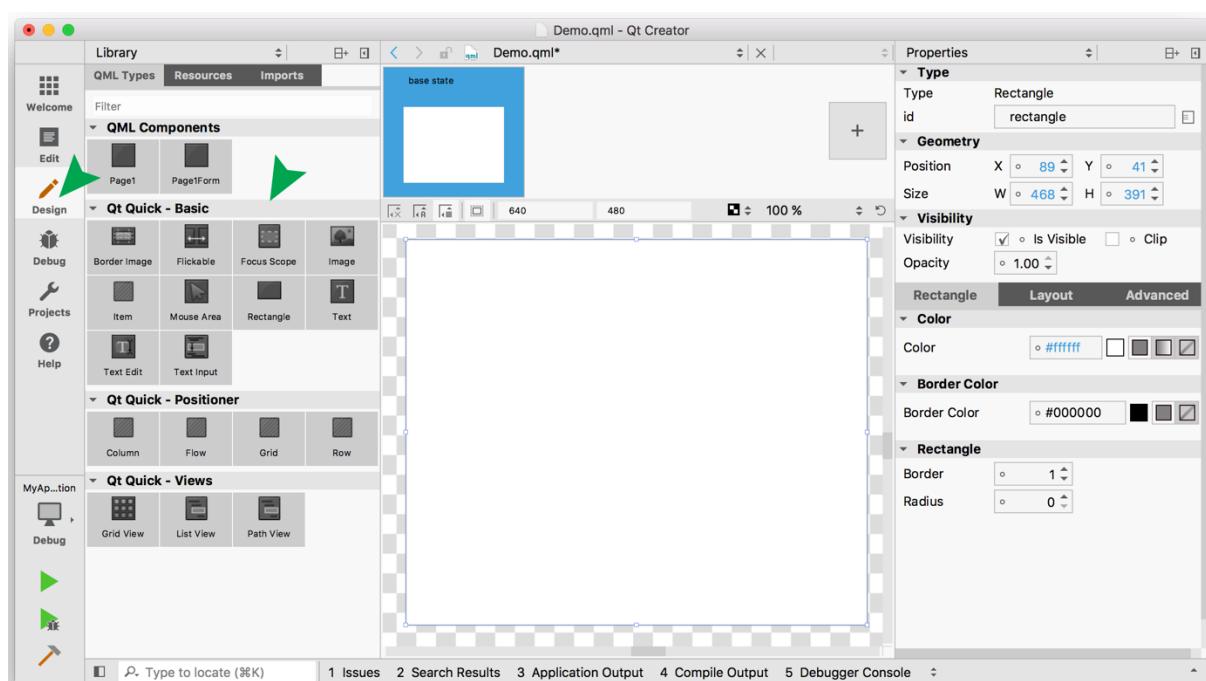
در QML انواع مختلفی برای طراحی رابط‌کاربری موجود است که بسیاری از آن‌ها خود دارای ویژگی‌های منحصر به فردی هستند که ترکیب هریک از آن‌ها با یکدیگر نتیجه‌ای بسیار عالی را رقم خواهد زد.

## معرفی انواع QML در فناوری Qt Quick

ماژول Qt Quick انواع مختلف گرافیکی را در خود جای داده است و دسترسی به آنها تحت اسناد QML امکان پذیر است. برای دسترسی و وارد کردن انواع مختلف و پایه QML مبتنی بر فناوری کیوت کوئیک از دستور زیر استفاده می‌کنیم:

```
import QtQuick 2.9
```

با وارد کردن این دستور انواع مختلفی در محیط طراحی (Designer) قابل مشاهده خواهد بود، که البته بسیاری از اعضای مرتبط به صورت مخفیانه در بخش ویرایشگر کد در دسترس قرار خواهد گرفت.



همانطور که مشاهده می‌کنید، با مراجعه به بخش Design دسترسی به انواع مختلف مانند مستقر کننده‌ها (Positioner)، نمایه‌ها (Views) و همچنین انواع پایه (Basic) صورت می‌گیرد.

با کشیدن و رها کردن هر یک از این انواع، می‌توان آن را بر روی فرم طراحی قرار داد و خاصیت‌های آن را بر اساس سلیقه، تغییر و توسعه داد. کافی است با دانش CSS، JavaScript و HTML5 نیز آشنا باشید.

تعدادی از انواع پایه وجود دارد که به صورت پیش فرض توسط زبان QML پشتیبانی می‌شوند.

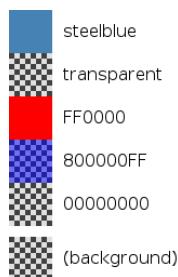
فراهم کننده مقدار رنگ از نوع RGB که با ارزش گذاری ARGB قابل استفاده است. برای مثال مقدار "red" و یا "#F23" است.	<b>color</b>
فراهم کننده نوع زمان برای مثال "YYYY-MM-DD"	<b>date</b>
فراهم کننده نوع فونت که بر پایه کلاس QFont قابلیت سفارش سازی را دارد.	<b>font</b>
ماتریکس ۴ در ۴ با ۴ ستون و ۴ ردیف.	<b>matrix4x4</b>
مشخص کننده مقدار محور x و y	<b>point</b>
مشخص کننده نوع چهارگانه توسط محورهای z, y, x و scalar	<b>quaternion</b>
نوع rect که به مقادیر محورهای y, x و عرض و ارتفاع اشاره دارد.	<b>rect</b>
ارزش با ویژگی عرض و طول	<b>size</b>
فراهم کننده نوع وکتور ۲ بعدی که دارای ویژگی مختصات x و y است.	<b>vector2d</b>
فراهم کننده نوع وکتور ۳ بعدی که دارای هر سه محور y, x و z است.	<b>vector3d</b>
فراهم کننده نوع وکتور ۴ بعدی که دارای محورهای z, y, x و w است.	<b>vector4d</b>

## نوع **color**

این نوع یک مقدار از رنگ با قالب ARGB را فراهم می‌کند. نوع رنگ به یک ARGB اشاره دارد. نحوه استفاده از آن به چند شیوه مختلف است.



- توسط نام یک رنگ از نوع SVG، مانند "lightsteelblue"، "green"، "red" و یا "lightblue"
- توسط یک کد هگزا دسیمال سه گانه "#RRGGBB" یا یک نوع چهار گانه "#AARRGGBB"



همانطور که می‌دانید طیف بسیار گسترده‌ای از رنگ‌ها، تحت سی‌پلاس‌پلاس و QML پشتیبانی می‌شود. در این میان شما می‌توانید رنگ دلخواه خود را تحت یک لیست داده‌ای برای کنترل QML ای خود ارسال کنید.

نمونه کد زیر شکل کلی پشتیبانی از رنگ را در QML نمایش می‌دهد.

```
Rectangle {
    color: "#FF0000"
    y: 80; width: 40; height: 40
}

Rectangle {
    color: "red"
    y: 160
    width: 40; height: 40
}

Rectangle {
    color: "#800000FF"
    y: 120; width: 40; height: 40
}

Rectangle {
    color: "#00000000"      // ARGB fully transparent
    y: 160
    width: 40; height: 40
}

Rectangle {
    color: "steelblue"
    width: 40; height: 40
}

Rectangle {
    color: "transparent"
    y: 40; width: 40; height: 40
}
```

یک نوع رنگ دارای خواص B, G, R و خواص A است. که نشانگر Red, Green, Blue و Alpha است. علاوه بر آن شامل hsvSaturatio, hsvHue و hslLightness است.

<https://www.w3.org/TR/SVG/types.html#ColorKeywords> :SVG لیست رنگ‌ها در قالب

## date نوع

یک نوع تاریخ در QML، به یک تاریخ اشاره می‌کند. برای ایجاد ارزش یک تاریخ، به صورت "YYYY-MM-DD" در قالب یک رشته مشخص می‌شود. شکل کلی آن به صورت زیر است:

### Text

```
{  
    text: Date("YYYY-MM-DD")  
}
```

## font نوع

مقدار فونت (قلم) بر اساس خواصی که در کلاس C++ با نام QFont موجود است دریافت می‌شود. نوع font به مقدار فونت با تمامی ویژگی‌ها در کلاس QFont اشاره دارد.

ویژگی‌هایی که بیشترین استفاده را دارند به صورت زیر آمده است:

- صفت font.family با نوع رشته (string)
  - صفت font.bold با نوع بولین (bool)
  - صفت font.italic با نوع بولین (bool)
  - صفت font.underline با نوع بولین (bool)
  - صفت font.pointSize با نوع صحیح اعشاری (real)
  - صفت font.pixelSize با نوع صحیح (int)
- نکته: هر دو صفت pointSize و pixelSize بر اساس اندازه پیکسل‌ها کار می‌کنند.
- همچنین ویژگی‌های زیر نیز در دسترس هستند:
- صفت font.weight با نوع شمارشی (enumeration)
  - صفت font.overline با نوع بولین (bool)
  - صفت font.strikeout با نوع بولین (bool)
- صفت font.capitalization با نوع شمارشی (enumeration)
- صفت font.letterSpacing با نوع صحیح اعشاری (real)

• صفت font.wordSpacing با نوع صحیح اعشاری (real)

شكل کلی استفاده از نوع قلم (font) به صورت زیر است:

```
Text { font.family: "Tahoma"; font.pointSize: 16; font.bold: true }
```

توجه داشته باشید که در زمان یکپارچه سازی با C++, هر مقدار از QFont از طرف سی‌پلاس‌پلاس به QML به طور خود کار تبدیل شده و ارسال می‌شود. همچنین این فرآیند به صورت برعکس نیز اتفاق می‌افتد. وزن یا ضخامت قلم از مقدار ۰ تا ۹۹ مشخص می‌شود. مقدار ۰ به معنای فوق سبک و مقدار ۹۹ به معنای بسیار سنگین و ضمین است. و طبق لیست زیر می‌توان از آن‌ها استفاده کرد:

0	Font.Thin
12	Font.ExtraLight
25	Font.Light
50	Font.Normal
57	Font.Medium
63	Font.DemiBold
75	Font.Bold
81	Font.ExtraBold
87	Font.Black

نکته: در صورتی که خاصیت font.bold مقدار دهی شده باشد خاصیت font.weight هیچ تاثیری بر روی محتوا نخواهد گذاشت. بنابراین، مثال فوق یک نمونه مناسب از این خاصیت است:

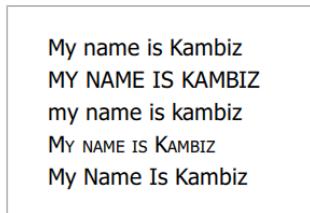
```
Text {  
    text : "My name is Kambiz"  
    font.family: "Tahoma";  
    font.pointSize: 16;  
    font.weight: Font.ExtraLight  
}
```

همچنین ویژگی‌های اختصاصی جهت اعمال حروف‌های بزرگ و کوچک به صورت زیر است:

بدون هیچ تغییری حالت بزرگ و کوچک سازی قلم اعمال می‌شود.	Font.MixedCase
باعث تغییر در متن شده و آن را به صورت حروف بزرگ نمایش می‌دهد.	Font.AllUppercase
باعث تغییر در متن شده و آن را به صورت حروف کوچک نمایش می‌دهد.	Font.AllLowercase
متن به طور کلی تغییر نخواهد کرد اما در سبک حروف کوچک نمایش داده خواهد شد.	Font.SmallCaps
بر روی متن تاثیر خواهد گذاشت، تمامی کلمات اول هریک از کلمه‌ها به صورت بزرگ نمایش داده خواهد شد.	Font.Capitalize

به عنوان مثال تکه کد زیر با به کارگیری خاصیت font.Capitalize ارائه شده است که نتیجه آن به صورت زیر است:

```
font.capitalization: Font.MixedCase  
font.capitalization: Font.AllUppercase  
font.capitalization: Font.AllLowercase  
font.capitalization: Font.SmallCaps  
font.capitalization: Font.Capitalize
```



## نوع matrix4x4

نوع ماتریس ۴ در ۴ چهار ردیف در چهار ستون را فراهم می‌کند. این نوع چهار در چهار ۱۶ مقدار دارد، هر کدام از طریق خاصیت‌های m11 و از طریق m44 در QML در سطر/ستون دسترسی دارند. مقادیر این نوع را می‌توان با استفاده از تابع () Qt.matrix4x4 تشکیل داد. هریک از ویژگی‌ها در ماتریس چهار در چهار به عنوان یک مقدار واقعی ذخیره می‌شود همچنین باید به این نکته توجه داشته باشید که دقت آن در معماری arm عادی بوده و در معماری x86 دو برابر است.

کد فوق ضرب یک ماتریس ۴ در ۴ را در QML مشخص می‌کند:

```
VAR A = QT.MATRIX4X4(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16);  
  
VAR B = QT.MATRIX4X4(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19);  
  
VAR C = A.TIMES(B);  
  
CONSOLE.LOG(C.TOSTRING());
```

نتیجهٔ خروجی :

```
qml: QMatrix4x4(  
120, 130, 140, 150, 280, 306, 332, 358, 440, 482, 524, 566, 600, 658, 716, 774  
)
```

## نوع point

یک مقدار با ویژگی‌های محور x و y که به ویژگی‌های نقاط مذکور اشاره دارد. شکل کلی آن به صورت زیر به صورت عادی و با استفاده از Qt.point مشخص شده است:

```
CUSTOMOBJECT { MYPOINTPROPERTY: "0,20" }  
  
CUSTOMOBJECT { MYPOINTPROPERTY: QT.POINT(0, 20) }
```

## نوع quaternion

نوعی چهار گانه که از محور x, y و z (محور اعداد) را پشتیبانی می‌کند. برای ایجاد یک مقدار چهار گانه، مشخص کردن آن به عنوان یک محور تحت "scalar" (محور اعداد) به صورت رشته و یا به صورت جدا گانه و تک تک تعریف می‌شود و یا توسط تابع `Qt.quaternion()` صورت می‌گیرد.

## نوع rect

یک مقدار با ویژگی‌های محور x و y و همچنین عرض و ارتفاع که به ویژگی‌های مذکور اشاره دارد.  
فرض کنید آیتمی به شکل زیر را در اختیار داریم که از ویژگی‌های عرض و ارتفاع برخوردار است:

```
Rectangle {  
    width: childrenRect.width  
    height: childrenRect.height  
  
    Rectangle { width: 100; height: 100 }  
}
```

حال می‌خواهیم با استفاده از نوع `rect` این ویژگی را برای شیء اختصاصی خودمان تخصیص دهیم.

```
CustomObject { myRectProperty: "50,50,100x100" }  
CustomObject { myRectProperty: Qt.rect(50, 50, 100, 100) }
```

## نوع size

یک نوع با ویژگی‌های عرض و ارتفاع، که به مقادیر عرض و ارتفاع شیء اشاره دارد.  
با فرض اینکه تصویری را به صورت زیر در اختیار داریم:

```
Image { id: image; source: "logo.png" }
```

حال در نظر داریم ابعاد ارتفاع و عرض آن را اختصاص دهیم :

```
Image { sourceSize: "100x100" }  
Image { sourceSize: Qt.size(100, 100) }
```

همانطور که مشاهده می‌کنید توسط نوع `size` می‌توان مقادیر را به دو شیوه ذکر شده به اشیاء اختصاص داد.

## نوع vector2d

یک نوع وکتور دو بعدی که ویژگی محورهای  $x$  و  $y$  را دارد. یک نوع وکتور دو بعدی دارای ویژگی های محورهای  $x$  و  $y$  است، در غیر اینصورت همانند نوع وکتور سه بعدی است. در ادامه در رابطه با ویژگی های نوع وکتور سه بعدی توضیح داده خواهد شد. برای ساخت یک وکتور دو بعدی، مشخص کردن آن از نوع رشته به صورت " $x,y$ " و یا مشخص کردن آن به صورت جدا گانه صورت می‌گیرد و یا توسط تابع `Qt.vector2d()` ایجاد می‌شود.

کد زیر مثالی از نحوه استفاده نوع وکتور ۲ بعدی است که با اختصاص دادن مقادیر در محورهای  $x$  و  $y$  توسط تابع `vector2d` فراهم شده است که نتیجه خروجی آن ۳۸ خواهد بود.

نکته: تابع `dotProduct` و مسائل مربوط به وکتور در کتابخانه های وکتور موجود هستند و در این کتاب به مباحث مرتبط به آن اشاره نمی‌شود.

```
var a = Qt.vector2d(1,2);
var b = Qt.vector2d(3,4);
var c = a.dotProduct(b);
console.log(c);
```

## نوع vector3d

نوع وکتور سه بعدی که ویژگی محورهای  $x$ ,  $y$  و  $z$  را دارا است. در ادامه در رابطه با ویژگی های نوع وکتور سه بعدی توضیح داده خواهد شد. برای ساخت یک وکتور سه بعدی، مشخص کردن مختصات آن از نوع رشته به صورت " $x,y,z$ " و یا به صورت جدا گانه صورت می‌گیرد و یا توسط تابع `Qt.vector3d()` ایجاد می‌شود.

کد زیر مثالی از نحوه استفاده نوع وکتور ۳ بعدی است که با اختصاص دادن مقادیر در محورهای  $x$  و  $y$  و  $z$  توسط تابع `vector3d` فراهم شده است که نتیجه خروجی آن  $(-6, 42, -27)$  `QVector3D` خواهد بود.

```
var a = Qt.vector3d(1,4,6);
var b = Qt.vector3d(8,5,6);
var c = a.crossProduct(b);
console.log(c.toString());
```

لازم است جهت استفاده های دیگر وکتور نوع ۳ بعدی مثال های زیر را در نظر داشته باشید:

```
Rotation { angle: 60; axis: "0,1,0" }
Rotation { angle: 60; axis: Qt.vector3d(0, 1, 0) }
Rotation { angle: 60; axis.x: 0; axis.y: 1; axis.z: 0 }
```

مثال های مذکور نشان دهنده این هستند که وکتور ۳ بعدی در یک نوع `Rotation` چگونه مورد استفاده قرار می‌گیرد.

## نوع vector4d

نوع وکتور چهار بعدی که ویژگی محورهای  $x, y, z$  و  $w$  را دارد. یک نوع وکتور چهار بعدی دارای ویژگی محورهای  $x$  و  $y$  و  $z$  و  $w$  است، در غیر اینصورت همانند نوع وکتور سه بعدی است. در ادامه در رابطه با ویژگی‌های نوع وکتور سه بعدی توضیح داده خواهد شد.

برای ساخت یک وکتور از نوع چهار بعدی، مشخص کردن آن از نوع رشته به صورت " $x,y,z,w$ " و یا مشخص کردن آن به صورت جدا گانه صورت می‌گیرد و یا توسط تابع `Qt.vector4d()` ایجاد می‌شود.

کد زیر مثالی از نحوه استفاده نوع وکتور ۴ بعدی است که با اختصاص دادن مقادیر در محورهای  $x, y, z$  و  $w$  توسط تابع `vector4d` فراهم شده است که نتیجه خروجی آن ۲۲۷ خواهد بود.

```
var a = Qt.vector4d(1,4,6,8);
var b = Qt.vector4d(3,9,10,16);
var c = a.dotProduct(b);
console.log(c);
```

## معرفی انواع اشیاء QML در Qt Quick

تمامی شیوه‌های ارائه شده توسط فناوری **Qt Quick** بر پایه نوع **Item** هستند که از کلاس **Qt Object** مشتق شده‌اند. اشیاء موجود در نوع QML توسعه مازول QML در دسترس قرار می‌گیرد که شامل اشیاء و اجزاء (کامپوننت) هستند. همچنین تنها زمانی در دسترس قرار می‌گیرند که شما مازول Qt Quick را فراخوانی کرده باشید.

فعال کننده دسترسی به آیتم‌ها	<b>Accessible</b>
حرکات تغییر دهنده لنگرهای (anchors)	<b>AnchorAnimation</b>
مشخص کننده حالت تغییر لنگرهای anchors از وضعیت یک آیتم.	<b>AnchorChanges</b>
نمایش دهنده آنیمیشن‌های متحرک ذخیره شده به عنوان یک سری از تصاویر ترسیم کننده اسپرایت (sprite) متحرک.	<b>AnimatedImage</b>
این گزینه، پایه و اساس تمامی آنیمیشن‌های QML است.	<b>AnimatedSprite</b>
فعال‌سازی دستی کنترل کننده آنیمیشن‌ها.	<b>Animation</b>
این گزینه، پایه و اساس همه آنیماتور (آنیمیشن)‌های QML محسوب می‌شود.	<b>AnimationController</b>
یک آنیمیشن پیشفرض برای تغییر وضعیت تعریف می‌کند.	<b>Animator</b>
محدوده یا مرزی را بر پایه یک تصویر ترسیم می‌کند.	<b>Behavior</b>
فراهم کننده زمینه‌سازی برای اشکال دو بعدی بر روی آیتمی در بوم.	<b>BorderImage</b>
یک بوم دو بعدی را برای ترسیم توسط جاوااسکریپت (JavaScript) فراهم می‌کند.	<b>Context2D</b>
یک بوم گرادیان را به عنوان رابط فراهم می‌کند.	<b>Canvas</b>
شامل داده‌های پیکسلی تصویر در RGBA است.	<b>CanvasGradient</b>
فراهم کننده ارسال و مدیریت دستورات و همچنین لیست کننده و دسترسی‌های لازم به یک جزء در هر یک از داده پیکسل‌های تصویر است.	<b>CanvasImageData</b>
جلوه‌های آنیمیشنی را هنگام تغییر ارزش - مقدار رنگ اعمال می‌کند.	<b>CanvasPixelArray</b>
فراهم کننده موقعیت زیر مجموعه‌ها در یک ستون است.	<b>ColorAnimation</b>
تعريف کننده انتبار سنج برای اعداد غیر صحیح.	<b>Column</b>
وظیفه مشخص‌سازی رویدادهای کشیدن و رها کردن را برای حرکت آیتم‌ها بر عهده دارد.	<b>DoubleValidator</b>
ارائه کننده اطلاعات مربوط به رویداد کشیده شدن (Drag).	<b>Drag</b>
وظیفه مشخص سازی کشیدن و رها کردن بر روی یک نقطه را بر عهده دارد.	<b>DragEvent</b>
ویژگی‌های ورودی صفحه کلید را فراهم می‌کند.	<b>DropArea</b>
سطحی از لایه‌ها را فراهم می‌کند که می‌تواند تکان دهنده باشد (لیز خوردن - حرکت ملایم - جهش پیدا کردن).	<b>EnterKey</b>
سطحی از لایه‌ها را فراهم می‌کند که می‌تواند یک تلنگر (ضربۀ) اساسی به لایه مورد نظر وارد کند. در واقع یک جهش شدید نسبت به مدل قبلی.	<b>Flickable</b>
تعیین کننده موقعیت‌های آیتم‌های فرزند - زیر آیتم‌ها در کنار هم قرار گرفته و در صورت لزوم آنها را بسته بندی خواهد کرد.	<b>Flipable</b>
ایجاد کننده صریح یک دامنه متمرکز است.	<b>Flow</b>
	<b>FocusScope</b>

اجازه می‌دهد که قلم مورد نظر بر اساس نام و یا آدرس URL بارگذاری شود.	<b>FontLoader</b>
معیارهای را برای فونت معین شونده فراهم می‌کند.	<b>FontMetrics</b>
یک گرادیان تعریف می‌کند.	<b>Gradient</b>
رنگی را در یک موقعیتی از گرادیان تعریف می‌کند.	<b>GradientStop</b>
تعیین موقعیت آیتم‌های فرزند در داخل یک شبکه توری - (Grid) را بر عهده دارد.	<b>Grid</b>
تعریف کنندهٔ تور با راس (مرکز) مرتب شده در یک شبکه (Grid)	<b>GridMesh</b>
مشخص کنندهٔ لایه‌ای از آیتم‌ها با نمای شبکه توری که توسط یک مدل فراهم می‌شود را فراهم می‌کند. یک لایه پر کاربرد در طراحی فرم محسوب می‌شود.	<b>GridView</b>
نمایش دهنده یک تصویر.	<b>Image</b>
تعریف کنندهٔ یک اعتبار سنج برای اعداد صحیح.	<b>IntValidator</b>
یک کنترل QML از نوع بصری و پایه است.	<b>Item</b>
نتایجی را شامل می‌شود که بر اساس صدا زده شدن از طرف یک آیتم مانند Item::grabToImage دریافت و آن را ذخیره می‌کند. برای مثال دریافت تصویری از فرم و تبدیل آن به نوع تصویری.	<b>ItemGrabResult</b>
ارائه دهندهٔ اطلاعات در بارهٔ یک رویداد کلیدی.	<b>KeyEvent</b>
پشتیبانی از یک کلید ناوبنی توسط کلیدهای جهت بر روی صفحه کلید یا دستگاه‌های دیگر را بر عهده دارد.	<b>KeyNavigation</b>
فرآیند دست زدن و لمس کردن آیتم‌ها را فراهم می‌کند.	<b>Keys</b>
مشخصه‌ای است که قابلیت انعکاس برعکس به صورت آینه را فراهم می‌کند. کاربرد بسیاری در راست چین و چپ چین کردن طراحی دارد.	<b>LayoutMirroring</b>
نمایش لیست مجموعه‌ای از آیتم‌ها را که تحت مدل فراهم می‌شود را پشتیبانی می‌کند.	<b>ListView</b>
وظیفه بارگذاری پویای کامپوننت و یا فایل‌های QML را بر عهده دارد.	<b>Loader</b>
روشی را برای اعمال انتقال ماتریس‌های $4 \times 4$ به یک آیتم فراهم می‌کند.	<b>Matrix4x4</b>
یک روش برای فعال‌سازی مدیریت رویدادهای ماوس است. رویداد لمسی تحت همین گزینه قابل استفاده است.	<b>MouseArea</b>
ارائه دهندهٔ اطلاعات در مورد رویدادهای ماوس.	<b>MouseEvent</b>
امکان لمس کردن چند نقطه را به صورت هم زمان فراهم می‌سازد.	<b>MultiPointTouchArea</b>
فراهم کنندهٔ تغییر و حرکت انیمیشن‌ها بر اساس مقادیر عددی.	<b>NumberAnimation</b>
اعمال انیمیشن برای خاصیت شفافیت (کدورت) را در یک آیتم فراهم می‌کند.	<b>OpacityAnimator</b>
اطلاعاتی را در رابطه اطلاعات متصل به یک اند نمودارهای صحنه‌ای ارائه می‌کند.	<b>GraphicInfo</b>
قابلیت اجرا شدن انیمیشن به صورت موازی را فراهم می‌سازد.	<b>ParallelAnimation</b>
انیمیشن‌ها به حالت مقادیر والد تغییر پیدا می‌کنند	<b>ParentAnimation</b>
مشخص می‌کند که چگونه یک آیتم تغییر حالت می‌دهد.	<b>ParentChange</b>

مسیری را برای استفاده از PathView تعریف می‌کند.	<b>Path</b>
یک آیتم را در امتداد یک مسیر به حرکت در می آورد.	<b>PathAnimation</b>
قوسی را با شعاع تعریف شده و مشخصی تعریف می‌کند.	<b>PathArc</b>
مشخص می‌کند که چگونه یک ویژگی-صفت را در یک موقعیت از مسیر باید اعمال کرد.	<b>PathAttribute</b>
تعریف یک منحنی مکعب بزرگ با دو نقطه کنترل	<b>PathCubic</b>
یک نقطه را در منحنی کتمول-رام (Catmull-Rom) تعریف می‌کند.	<b>PathCurve</b>
نوع مسیر پایه را مشخص می‌کند.	<b>PathElement</b>
مشخص می‌کند که چگونه به صورت دستی باید در امتداد یک مسیر حرکت کرد.	<b>PathInterpolator</b>
یک خط مستقیم - راست تعریف می‌کند.	<b>PathLine</b>
یک مسیر تعبیر شده - از قبل مشخص شده را دستکاری می‌کند.	<b>PathPercent</b>
یک مکعب درجه دوم با یک نقطه کنترل.	<b>PathQuad</b>
یک مسیر را با استفاده از یک مسیر رشته‌ای از نوع SVG را تعریف می‌کند.	<b>PathSvg</b>
یک سری از اقلام مدل‌های ارائه شده را بر روی یک مسیر قرار می‌دهد.	<b>PathView</b>
یک مکث را برای یک حرکتی از انیمیشن فراهم می‌کند.	<b>PauseAnimation</b>
یک محدوده‌ای به عنوان مکان‌های ناچیز و کوچک فعال می‌کند.	<b>PinchArea</b>
برای مشخص سازی اطلاعاتی درباره رویداد ناچیز استفاده می‌شود.	<b>PinchEvent</b>
خواص ضمیمه شده‌ای را فراهم می‌کند که شامل جزئیاتی در مورد محل استقرار یک آیتم است.	<b>Positioner</b>
تغییرات فوری را در حین تغییر حرکات (انیمیشن) مشخص می‌کند.	<b>PropertyAction</b>
انیمیشن‌ها در حالت مقادیر والد تغییر پیدا می‌کنند.	<b>PropertyAnimation</b>
اتصالات ویژگی جدید یا مقادیر یک حالت را توصیف می‌کند.	<b>PropertyChanges</b>
یک مستطیل توپر را با یک مرز اختیاری رسم می‌کند.	<b>Rectangle</b>
یک اعتبار سنج رشته‌ای را فراهم می‌کند.	<b>RegExpValidator</b>
با استفاده از مدل فراهم شده، تعدادی از مولفه‌های مبتنی بر آیتم را معرفی می‌کند.	<b>Repeater</b>
یک روش برای چرخاندن یک آیتم را فراهم می‌کند.	<b>Rotation</b>
انیمیشن‌ها در حالت مقادیر چرخش تغییر پیدا می‌کنند.	<b>RotationAnimation</b>
نوع انیمیشن را در چرخش یک آیتم مشخص می‌کند.	<b>RotationAnimator</b>
موقعیت فرزند - زیر مجموعه‌های خودش را در یک ردیف مشخص می‌کند.	<b>Row</b>
روشی را برای مقایسه یک آیتم فراهم می‌کند.	<b>Scale</b>
نوع انیمیشن را بر اساس ضریب مقیاس یک آیتم مشخص می‌کند.	<b>ScaleAnimator</b>

اسکریپت‌هایی را تعریف می‌کند که در طول اجرای یک اینیمیشن اجرا خواهد شد.	<b>ScriptAction</b>
اجازه می‌دهد تا اینیمیشن‌ها به ترتیب اجرا شوند.	<b>SequentialAnimation</b>
اعمال کننده سایه‌های سفارشی به یک مستطیل.	<b>ShaderEffect</b>
ارائه یک آیتم Qt Quick در یک بافت و نمایش آن.	<b>ShaderEffectSource</b>
میانبرهای کلیدی را فراهم می‌کند.	<b>Shortcut</b>
به یک ویژگی اجازه می‌دهد که به آرامی یک مقدار را دنبال کند.	<b>SmoothedAnimation</b>
اجازه می‌دهد که یک ویژگی یک مقدار را مانند سیر تکاملی فصل‌ها دنبال کند.	<b>SpringAnimation</b>
دسترسی به پالت کیوت را فراهم می‌کند.	<b>SystemPalette</b>
مشخص می‌کند که چطور یک متن فرمت شده را می‌توان به یک صحنه اضافه کرد.	<b>Text</b>
چند خط از متن فرمت شده‌ی قابل ویرایش را نمایش می‌دهد.	<b>TextEdit</b>
خط قابل ویرایش از یک متن را نشان می‌دهد.	<b>TextInput</b>
معیارهایی را برای متن و فونت داده شده فراهم می‌کند.	<b>TextMetrics</b>
یک نقطه‌ی لمسی را در یک MultiPointTouchArea توصیف می‌کند.	<b>TouchPoint</b>
برای مشخص کردن تغییرات پیشرفت‌های بر روی آیتم‌ها به کار می‌رود.	<b>Transform</b>
انیمیشن‌های متحرک را در طول تغییر حالت تعریف می‌کند.	<b>Transition</b>
روشی را برای انتقال یک آیتم بدون تغییر مختصات x و y آن تعریف می‌کند.	<b>Translate</b>
نوع اینیمیشن را بر اساس سایه‌های یکنواخت مشخص می‌کند.	<b>UniformAnimator</b>
انیمیشن‌ها در مقادیر QVector3d تغییر می‌کنند.	<b>Vector3dAnimation</b>
آیتم‌هایی که تحت یک نمای خاص هستند مشخص می‌کند.	<b>ViewTransition</b>
اطلاعاتی را در مورد رویداد چرخش ماوس فراهم می‌کند.	<b>WheelEvent</b>
نوع اینیمیشن را بر اساس مختصات x یک آیتم مشخص می‌کند.	<b>XAnimator</b>
نوع اینیمیشن را بر اساس مختصات y یک آیتم مشخص می‌کند.	<b>YAnimator</b>

## Accessible نوع

این کلاس بخشی از دسترسی (Accessible) برای پروژه‌های تحت Qt Quick است. آیتم‌هایی که کاربر با آن‌ها در تعامل است یا اطلاعاتی را به کاربر می‌دهد، نیاز به افشای اطلاعات آن‌ها در چهارچوبی دسترسی پذیر است. سپس ابزارهای کمکی می‌توانند از این اطلاعات برای فعال کردن کاربران جهت تعامل با اپلیکیشن‌ها با روش‌های مختلف استفاده کنند. مهمترین ویژگی‌ها عبارتند از : name، description و role

زیر نمونه‌ای جهت اجرای یک دکمه ساده است:

```
Rectangle {
    id: myButton
```

```

Text {
    id: label
    text: "next"
}

Accessible.role: Accessible.Button
Accessible.name: label.text
Accessible.description: "shows the next page"
Accessible.onPressAction: {
    // do a button click
}
}

```

کد فوق مشخص می‌کند که شیء myButton قابلیت‌های name، description و role خود را برای دسترسی ابزارهای کمکی فعال نماید. جهت اطلاعات بیشتر در رابطه با این قابلیت‌ها اینگونه توضیح داده می‌شود: فرض کنید قرار است سیستم از ابزارهای کمکی استفاده کند، ابزاری برای راحتی کار در استفاده کامپیوتر توسط افراد نابینا!

در این صورت ابزارهای کمکی بر اساس مقادیر موجود در بخش طراحی دسترسی‌های لازم برای خواندن متون موجود نیاز دارند که توسط نوع Accessible این امکان را می‌توان فراهم کرد.

## AnchorAnimation نوع

نوع لنگر انیمیشن (AnchorAnimation)، انیمیشنی را برای تغییر وضعیت یک لنگر (anchor) تعیین می‌کند که شکل آن به صورت زیر است :

```
AnchorAnimation { duration: 1000 }
```

نوع	صفت
int	duration
enumeration	easy.type
real	easing.amplitude
real	easing.overshoot
real	easing.period
list<Item>	targets

ویژگی duration مدت زمان انیمیشن را مشخص می‌کند که بر اساس واحد میلی ثانیه است و به صورت پیش فرض مقدار ۲۵۰ است. همچنین یک مشخص کننده برای حالت منحنی در موقع اجرای انیمیشن است که برای استفاده و حالت دادن به انیمیشن با ویژگی easy.type ترکیب می‌شود.

ساختار آن به صورت زیر است :

```
AnchorAnimation { easing.type: Easing.InOutQuad }
```

کد زیر نشان می‌دهد که چگونه یک انیمیشن می‌تواند به وضعیت سمت راست لنگر شده یک شیء مستطیل اختصاص یابد:

```
Item {
    id: container
    width: 200; height: 200

    Rectangle {
        id: myRect
        width: 100; height: 100
        color: "red"
    }

    states: State {
        name: "reanchored"
        AnchorChanges { target: myRect; anchors.right: container.right }
    }
}

transitions: Transition {
    // smoothly reanchor myRect and move into new position
    AnchorAnimation { easing.type: Easing.InOutQuad }
}

Component.onCompleted: container.state = "reanchored"
```

## نوع AnchorChanges

لنگر انیمیشن (AnchorChanges) برای تغییر وضعیت (حالت) لنگرهای یک آیتم به کار می رود.

شکل کلی آن به صورت زیر است :

```
AnchorChanges {  
    target: myRect  
    anchors.right: window.right  
    anchors.left: window.left  
}
```

در زیر مثالی از نحوه عملکرد آن ارائه شده است که نتیجه آن در ادامه قابل مشاهده است:

```
Rectangle {  
    id: window  
    width: 256; height: 256  
    color: "orange"  
  
    Rectangle { id: myRect; width: 100; height: 100; color: "green" }  
  
    states: State {  
        name: "reanchored"  
  
        AnchorChanges {  
            target: myRect  
            anchors.right: window.right  
            anchors.left: window.left  
        }  
  
        PropertyChanges {  
            target: myRect  
            anchors.rightMargin: 25  
            anchors.leftMargin: 25  
        }  
    }  
  
    MouseArea { anchors.fill: parent; onClicked: window.state = "reanchored" }  
}
```



حال با کلیک بر روی ناحیه سبز رنگ، وضعیت شیء سبز رنگ بر اساس لنگرهای تعریف شده به صورت تصویر دوم تغییر خواهد کرد.

## نوع AnimatedImage

نوع تصویر متحرک (AnimatedImage)، یک نوع توسعه یافته با استفاده از تصویر است؛ با استفاده از ویژگی نوع عکس، که در این شیء کنترلی وجود دارد را می‌توانید تصاویری با قالب Gif را اجرا کنید. شکل کلی آن به صورت زیر است :

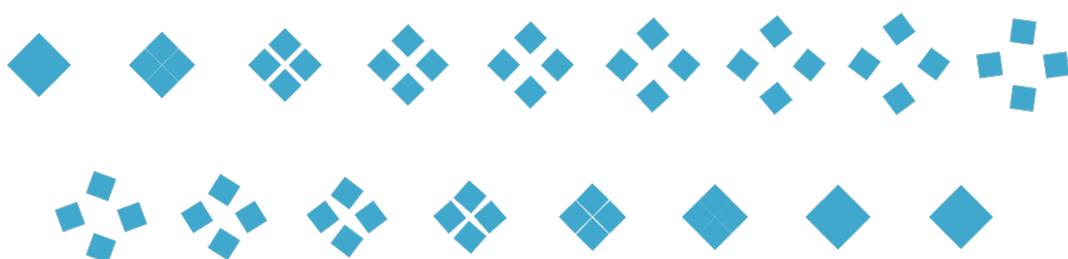
```
AnimatedImage { id: animation; source: "animation.gif" }
```

اطلاعات مرتبط با فریم فعلی و مجموع آن از یک انیمیشن را می‌توانید با استفاده از خواص currentFrame و frameCount به دست آورید.

نوع	صفت
int	currentFrame
int	frameCount
bool	paused
bool	playing
url	source

نمونه کد زیر مثالی از نحوه استفاده از یک فایل gif. را فراهم می‌کند:

```
AnimatedImage {  
    anchors.centerIn: parent  
    id: animation;  
    source: "qrc:/progress.gif"  
}
```



خاصیت source مسیر مرتبط با فایل gif. را دریافت و نمایش می‌دهد.

## نوع AnimatedSprite

این نوع تولید و کنترل تصویر متحرک را بر عهده دارد. همانند کنترل قبلی است، اما اجازه این را می‌دهد تا یک تصویر ثابت و غیر متحرک را به صورت انیمیشن نمایش دهید. برای مثال می‌توانید سرعت نمایش و تحرک یک انیمیشن تصویری را بر روی یک نرخ مورد نظر ثابت کرده و یا آن را به صورت دستی پیش ببرید.

اطلاعات مرتبط با فریم فعلی و مجموع آن از یک انیمیشن را می‌توانید با استفاده از خواص `frameCount` و `currentFrame` به دست آورید.

نوع	صفت
int	<code>currentFrame</code>
int	<code>frameCount</code>
int	<code>frameDuration</code>
int	<code>frameHeight</code>
qreal	<code>frameRate</code>
bool	<code>frameSync</code>
int	<code>frameWidth</code>
int	<code>frameX</code>
int	<code>frameY</code>
bool	<code>interpolate</code>
int	<code>loops</code>
bool	<code>paused</code>
bool	<code>reserves</code>
bool	<code>running</code>
url	<code>source</code>

کد زیر مثالی از نحوه استفاده از یک فایل png. را فراهم می‌کند تا بر اساس مشخصه‌های تعریف شده به صورت انیمیشن نمایش داده شود.

```
AnimatedSprite {
    anchors.centerIn: parent
    id: animation;
    source: "qrc:/marker.png"
    frameHeight:128
    frameWidth:128
    width:128
    height:128
    frameDuration: 100;
    frameCount: 10;
}
```



کد مربوطه با دریافت مسیر از فایل تصویر قادر است آن را تحت خاصیت‌هایی که به آن اختصاص و در آن تعریف شده است نمایان سازد. برایمثال صفت frameDuration برابر است با ۱۰۰ این بدین معنی است که در ۱۰۰ میلی ثانیه وضعیت تصویر تغییر خواهد کرد. گزینه frameCount برابر است با ۱۰ و به تعداد ۱۰ بار فاصله بین فریم اعمال خواهد شد. هرچقدر این مقدار پایین باشد، سرعت اجرا بیشتر و هرچقدر بالا باشد سرعت اجرا پایین‌تر خواهد بود.

نکته: جهت نمایان شدن تصویر خاصیت‌های frameWidth و frameHeight باید برابر باشند با اندازه اصلی فایل تصویر که در اینجا ما از یک فایل ۱۲۸ در ۱۲۸ پیکسل استفاده کرده‌ایم.

در صورتی که نیاز باشد تنها در تعداد دفعات محدودی تصویر مورد نظر اجرا شود کافی است از خاصیت loops استفاده کنید. برای مثال با وارد کردم مقدار ۳ به این صفت در ۳ مرحله تصویر شما نمایان خواهد شد.

## نوع Animation

این یک نوع پایه اینیمیشن (Animation) برای تمامی اینیمیشن‌ها در QML است، بر خلاف دیگر انواع نمی‌توان در داخل یک کنترل به عنوان یک خاصیت فرزند مورد استفاده قرار داد. اینکه نوع اینیمیشن به تنها ی نمی‌تواند مورد استفاده قرار بگیرد، نمونه کد زیر استفاده آن را همراه با یک شیء مستطیل فراهم آورده است.

نوع	صفت
int	alwaysRunToEnd
bool	loops
bool	paused
توضیح عملکرد سیگنال	سیگنال
این سیگنال زمانی ساطع می‌شود که اینیمیشن آغاز شود و کنترل کننده آن onStarteds است.	started
این سیگنال زمانی ساطع می‌شود که اینیمیشن تمام شود. و کنترل کننده آن onStoppeds است.	stopped
توضیح عملکرد تابع	عملکرد
زمانی که اینیمیشن تمام می‌شود، مقادیر به خاصیت نهایی خود می‌رسند. در صورتی که اینیمیشن در حال اجرا نباشد صدا زدن این روش بی تاثیر خواهد بود. بعد از به اتمام رسیدن اینیمیشن این خاصیت به صورت false تحت کنترل کننده complete تغییر خواهد یافت. و بر خلاف stop() گزینه complete() سریعاً اینیمیشن را به پایان می‌رساند.	complete
انیمیشن را در حالت توقف نگه می‌دارد.	pause
انیمیشن را مجددآ آغاز می‌کند.	restart
انیمیشن متوقف شده را به حالت عادی بر می‌گرداند.	resume
انیمیشن را آغاز می‌کند.	start
انیمیشن را به اتمام می‌رساند.	stop

```

Rectangle {

    width: 100; height: 100; color: "green"

    RotationAnimation on rotation {

        from: 0

        to: 360

        duration: 1000

    }

}

```

کد فوق با استفاده از خاصیت انیمیشن یک شکل مستطیل را از مدار ۰ تا ۳۶۰ درجه نمایش می‌دهد.

## نوع AnimationController

در حالت عادی انیمیشن‌ها توسط یک تایمر (زمان) داخلی هدایت می‌شود. در این میان کنترل کننده انیمیشن ([AnimationController](#)) اجازه می‌دهد تا وضعیت انیمیشن تحت مقادیر در حال پیشرفت هدایت شود.

## نوع Animator

انواع انیماتورها نوع خاصی از انیمیشن هستند که به طور مستقیم بر روی نمودارهای صحنه‌ای تحت Qt Quick عمل می‌کنند. این در حالتی است که حتی در زمانی که یک پروسه جهت صحنه‌های انیمیشنی در حال اجرا قرار بگیرد انیمیشن مورد نظر تحت این نوع می‌تواند وارد عمل شود. البته بخشی از این قابلیت موازی ساز است که تحت این نوع می‌تواند اعمال شود.

```

Rectangle {

    id: mixBox

    width: 50

    height: 50

}

ParallelAnimation {

    ColorAnimation {

        target: mixBox

        property: "color"

        from: "red"
    }
}

```

```

        to: "green";
        duration: 1000
    }

    ScaleAnimator {
        target: mixBox
        from: 2
        to: 1
        duration: 1000
    }
    running: true
}
}

```

در کد فوق اگر دقت کنید `ScaleAnimator` با در زمانی وارد عمل می‌شود که رنگ شیء مستطیل در حال تغییر است. در این زمان تحت این نوع وارد شده حالت مقایسی مستطیل مورد نظر تغییر خواهد کرد. در صفحات بعدی در رابطه با انواع `Animator` ها توضیحات داده خواهد شد.

## نوع Behavior

یک رفتار به صورت پیش فرض برای انیمیشن تعریف می‌شود. تا در زمان تغییر مقادیر بر اساس این رفتار تعریف شده عمل کند. برای مثال، رفتار زیر یک انیمیشن شمارشی را در زمان تغییر مقادیر مرتبط با عرض مستطیل را فراهم می‌کند.

زمانی که در ناحیه ماوس کلیک شود عرض آن بر اساس رفتار تعریف شده تغییر خواهد کرد.

```

Rectangle {
    id: rect
    width: 150; height: 150
    color: "green"

    Behavior on width {
        NumberAnimation { duration: 1000 }

    }

    MouseArea {
        anchors.fill: parent
        onClicked: rect.width = 100
    }
}

```

```
    }  
}  
}
```

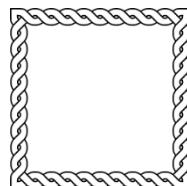
همانطور که مشخص است رفتار تعریف شده بر اساس نوع شمارشی مشخص شده است. توجه داشته باشید زمانی از نوع شمارشی استفاده می‌شود که قرار است ارزش یک مقدار عددی تغییر پیدا کند. برای مثال در این نمونه مقدار عرض که برابر است با ۱۵۰ موقع کلیک شدن به مقدار ۱۰۰ تغییر پیدا می‌کند. در صورتی که از رفتار مشخصی استفاده نشود این حالت بدون هیچ جلوه‌ی خاصی تغییر خواهد کرد.

## نوع **BorderImage**

نوع حاشیه تصویر (**BorderImage**) در ساخت حاشیه خارج از تصویر بر اساس مقایس یا بخشی از کاشی در هر قسمت از تصویر اعمال می‌شود. در واقع نوعی برای سفارشی سازی تصاویر حاشیه ساز است. ساختار آن به صورت زیر است:

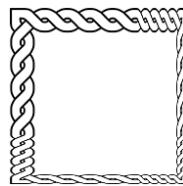
```
BorderImage {  
    source: "qrc:/BorderImage.png"  
    width: 256; height: 256  
}
```

به عنوان مثال نمونه تصویر زیر را در نظر بگیرید:



حال با استفاده از کد زیر جهت تغییر در حاشیه آن به صورت زیر عمل خواهیم کرد:

```
BorderImage {  
    source: "qrc:/BorderImage.png"  
    width: 180; height: 180  
    border { left: 130; top: 130; right: 130; bottom: 130 }  
    horizontalTileMode: BorderImage.Repeat  
    verticalTileMode: BorderImage.Repeat  
    anchors.centerIn: parent  
}
```



با توجه به اعمال تغییرات توسط خاصیت `border` تصویر فوق تغییر یافته است.

## نوع Context2D

هدف شیء `Context2D` این است که با روش `getContext()` برای بوم (`Canvas`) ایجاد شود. ساختار آن به صورت زیر است :

```
Canvas {  
    id:canvas  
    onPaint:{  
        var ctx = canvas.getContext('2d');  
        //...  
    }  
}
```

رابطهای برنامه‌نویسی `Context2D` شرایط و روش‌هایی مورد نیاز را جهت تولید تصویر ارائه می‌کند. نمونه زیر نشان می‌دهد که چگونه در یک متغیر `context` ذخیره می‌شود.

```
var context = mycanvas.getContext("2d")
```

## Canvas نوع

آیتم بوم (`Canvas`) اجازه می‌دهد تا خطوط راست و منحنی را رسم نمایید. اشکال ساده و پیچیده، نمودارها، و تصاویر گرافیکی که اشاره شده‌اند. همچنین می‌تواند متن، رنگ‌ها، سایه‌زنی، شبیه‌الگوها را اضافه کنید. این آیتم انجام عملیات مرتبط با پیکسل‌های سطح پایین را فراهم می‌کند. فراموش نکنید که خروجی بوم (`Canvas`) به صورت یک فایل تصویر یا سریالیز شده در یک مسیر (URL) صورت می‌گیرد.

تولید تصاویر تحت `Canvas` با استفاده از `Context2D` بوده و معمولاً نتایج آن به عنوان یک سیگنال رسم می‌شود. برای مثال برای طراحی یک ناحیه به صورت مستطیل تحت ارتفاع و عرض آن به صورت زیر عمل می‌کنیم.

```
Canvas {  
    id: mycanvas  
    width: 100  
    height: 100  
  
    onPaint: {
```

```

    var ctx = getContext("2d");
    ctx.fillStyle = Qt.rgba(1, 0, 0, 1);
    ctx.fillRect(0, 0, width, height);
}

}

```

جهت رسم طرح دو بعدی می بایست مقدار رشته از "2d" رنگ مربوطه را اعمال کرده و در نهایت `fillRect` مختصات و ابعاد مربوطه را اعمال می کند.

## CanvasGradient نوع

یک رنگ در حالت شبیه را برای Canvas می سازد.

برای طراحی یک ناحیه به صورت مستطیل تحت ارتفاع و عرض آن همراه با یک شکل به صورت زیر عمل می کنیم.

```

Canvas {
    id: mycanvas
    width: 100
    height: 100

    onPaint: {

        var ctx = getContext("2d")
        var gradient = ctx.createLinearGradient(100, 0, 100, 200)
        gradient.addColorStop(0, "red")
        gradient.addColorStop(0.5, "orange")
        ctx.fillStyle = gradient
        ctx.fillRect(0, 0, 200, 200)

    }
}

```

## CanvasImageData نوع

این شیء (CanvasImageData) ابعاد واقعی از تصویر را در خود ذخیره می کند.

## CanvasPixelArray نوع

هدف فراهم کردن دستورات برای شیء است این نوع می تواند به عنوان یک آرایه در جاوا اسکریپت در دسترس قرار بگیرد.

## ColorAnimation نوع

نوع ColorAnimation یک ویژگی اختصاصی تخصیص یافته شده برای تعریف یک انیمیشن است. این ویژگی موقع تغییر رنگ اعمال می شود. ساختار آن به صورت زیر است:

```

ColorAnimation on color {

    to: "red";
    duration: 1000;

}

```

جهت استفاده از این خاصیت در بین یک شیء می‌بایست بعد از نام آن کلمه کلیدی **on** را اعمال و سپس هدف تغییر را مرتبط با آن وارد می‌کنیم. که ویژگی مربوطه در این نوع رنگ خواهد بود که با کلمه **color** مشخص می‌شود.

نوع	صفت
int	<b>from</b>
int	<b>to</b>

```
Rectangle {
    width: 100; height: 100
    color: "green"

    ColorAnimation on color {
        to: "red"; duration: 1000;
    }
}
```

در کد فوق توسط خاصیت **ColorAnimation** رنگ سایز مستطیل را به رنگ قرمز در طی بازه زمانی ۱ ثانیه تغییر می‌دهد. با استفاده از خاصیت **to** مشخص شده است که رنگ مستطیل از چیزی که است به رنگ اشاره شده تغییر یابد. در صورتی که از صفت **from** استفاده شود هنگام اجرای این صفت رنگ اولیه تغییر خواهد یافت.

## نوع **Column**

نوعی برای اعتبار سنجی انواع غیر عدد صحیح است. ساختار آن به صورت زیر است :

```
Column {...}
```

نوع	صفت
Transition	<b>add</b>
real	<b>bottomPadding</b>
real	<b>leftPadding</b>
Transition	<b>move</b>
real	<b>padding</b>
Transition	<b>populate</b>
real	<b>rightPadding</b>
real	<b>spacing</b>
real	<b>topPadding</b>

نمونه کد زیر سه رنگ اصلی در ابعاد ۱۰۰ در ۱۰۰ پیکسل را در یک ستون با فاصله ۵ پیکسل از یکدیگر نمایش می‌دهد.

```
Column {
```

```

spacing: 5

Rectangle { color: "red"; width: 100; height: 100 }

Rectangle { color: "green"; width: 100; height: 100 }

Rectangle { color: "blue"; width: 100; height: 100 }

}

```

نتیجه آن به صورت زیر خواهد بود :



## DoubleValidator نوع

نوعی برای اعتبار سنجی انواع غیر عدد صحیح است، توسط این اعتبار سنج زمانی ورودی پذیرفته می شود که قالب و بازه ورودی آن صحیح باشد.  
شکل کلی آن به صورت زیر است :

```

DoubleValidator {

    bottom: -127.9;
    top: 127.9;
    decimals: 1;

}

```

فرض کنید نیاز است در یک فیلد متن مقدار خاصیت ورودی آن را به مقدار صحیح اعشاری محدود سازید، در این صورت شکل کلی کد به صورت زیر خواهد بود:

```

TextField {

    text: "0.0"

    validator: DoubleValidator {

        bottom: -127.9;
        top: 127.9;
        decimals: 1;
    }
}

```

## Drag نوع

با استفاده از خاصیت کشیدن (Drag) زمانی که خاصیت کشیده شدن بر روی یک آیتم فعال باشد، هر تغییراتی که در موقعیت آن رخ می‌دهد یک رویدادی را توسط کشیده شدن به کنترل DropArea ارسال خواهد کرد. اگر لازم باشد در فرم برنامه خود ناحیه‌ای را ایجاد کنید و بر روی آن آیتم‌های خود را جابجا نمایید این خاصیت بسیار کار آمد خواهد بود. همچنین آیتم‌های دیگر نیز می‌توانند این رویدادها را مدیریت نموده و رویدادهای آن را دریافت نمایند.

البته فراموش نکنید که این خاصیت به تنها یک قابل استفاده نیست، می‌بایست توسط نوع وابسته به آن یعنی ناحیه دریافت کننده رویدادها (DropArea) ترکیب شود. شکل کلی این خاصیت به صورت زیر است :

```
Drag.active: dragArea.drag.active  
Drag.hotSpot.x: 10  
Drag.hotSpot.y: 10
```

نوع	صفت
bool	active
enumeration	dragType
QPointF	hotSpot
QUrl	imageSource
stringlist	keys
stringlist	mimeData
enumeration	proposedAction
object	source
flags	supportedActions
object	target
توضیح عملکرد سیگنال	سیگنال
این سیگنال زمانی ساطع می‌شود که حالت کشیده شدن یک آیتم به اتمام رسیده باشد.	dragFinished
این سیگنال زمانی ساطع می‌شود که حالت کشیده شدن یک آیتم آغاز شده باشد.	dragStarted
توضیح عملکرد تابع	عملکرد
مراحل کشیده شدن را به پایان می‌رساند.	cancel
مراحل کشیده شدن را بعد از رها شدن آیتم به پایان می‌رساند.	drop
مراحل کشیده شدن را با آغاز کشیدن آیتم به شروع می‌کند.	start
رویدادهای کشیده شدن را ارسال می‌کند.	startDrag

در کد فوق dragArea نام ناحیه‌ای است که قرار است اطلاعات دریافتی توسط خاصیت Drag بر روی آن دریافت شود. کد زیر نمونه کاملی از نحوه استفاده این خاصیت همراه با ناحیه وابسته به آن را نمایش می‌دهد :

```
Item {
```

```

width: 200; height: 200
DropArea {
    x: 75; y: 75

    width: 50; height: 50
    Rectangle {
        anchors.fill: parent
        color: "green"

        visible: parent.containsDrag
    }
}

Rectangle {
    x: 10; y: 10
    width: 20; height: 20
    color: "red"
    Drag.active: dragArea.drag.active
    Drag.hotSpot.x: 10
    Drag.hotSpot.y: 10
    MouseArea {
        id: dragArea
        anchors.fill: parent
        drag.target: parent
    }
}
}

```

همانطور که مشاهده می‌کنید خاصیت active در ویژگی‌های آیتم مستطیل فعال شده و مختصات x همراه y عنوان شده است. از طرف دیگر با تغییر موقعیت آیتم در مختصات محور x و y در داخل DragArea رنگ زمینه شیء موجود در آن بخش به سبز تغییر خواهد کرد.

## DragEvent نوع

مختصات به دست آمده از محورهای  $x$  و  $y$  که حاصل از رویداد کشیده شدن و همچنین کلیدهای منابع رویداد را شناسایی می‌کند.

## DropArea نوع

آیتم (DropArea) یک آیتم نامرئی است و تمامی وقایعی را که آیتم‌های دیگر موقع کشیده و رها شدن منتشر می‌کنند را دریافت می‌کند.

## EnterKey نوع

نوع وابسته به خاصیت استفاده شده برای دستکاری ظاهر و رفتار کلیدهای وارد شده صفحه کلید بر روی صفحه نمایش است.

توضیحات	ثابت‌ها
به طور پیش فرض کلید Enter است. این می‌تواند به صورت یک دکمه برای پذیرش ورودی و یا بستن صفحه کلید و یا برای وارد شدن به یک خط جدید در ورودی متن با خاصیت چند خطی را فراهم کند.	Qt.EnterKeyDefault
دکمه‌ای را نمایش می‌دهد که یک خط جدید را درج می‌کند.	Qt.EnterKeyReturn
دکمه‌ای را با عنوان انجام شده "Done" نمایش می‌دهد. به طور معمول، صفحه کلید انتظار بسته شدن را موقع فشرده شدن کلید را دارد.	Qt.EnterKeyDone
نمایش یک دکمه با عنوان "Go"، به طور معمول در یک نوار آدرس جهت وارد کردن یک آدرس مورد استفاده قرار می‌گیرد.	Qt.EnterKeyGo
یک دکمه با عنوان "Send" را نمایش می‌دهد.	Qt.EnterKeySend
یک دکمه با عنوان "Search" را نمایش می‌دهد.	Qt.EnterKeySearch
یک دکمه با عنوان "Next" را نمایش می‌دهد.	Qt.EnterKeyNext
یک دکمه با عنوان "Previous" را نمایش می‌دهد.	Qt.EnterKeyPrevious

## Flickable نوع

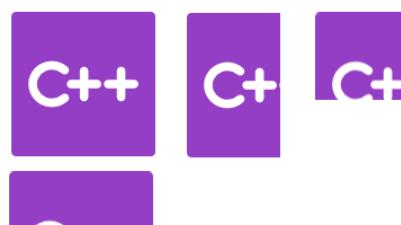
آیتم Flickable امکان کشیده شدن آیتم‌های فرزند خود را در روی یک سطح از فرم فراهم می‌کند. این خاصیت برای شکل دادن به آیتم‌ها طراحی شده است که برای استفاده از آیتم‌های GridView و ListView مناسب است. در رابطهای کاربری به سبک سنتی، ناحیه نمایش با استفاده از کنترل‌های اسکرول کننده فراهم می‌شود. حال در آیتم Flickable امکان نگه داشتن بر روی محیط و حرکت به اطراف این امکان را فراهم می‌سازد که به جای اسکرول شدن از خاصیت گرفتن و حرکت دادن به اطراف بازه نمایش را تغییر دهد. این امکان بیشتر در رابطهای کاربری لمسی به کار می‌گیرد.

رود که بیشتر در پلتفرم‌های موبایل کاربردی و مهم است. توجه داشته باشید که این کنترل به صورت خودکار محتوای خود را احاطه نمی‌کند. یا به طور کلی به قالب محتوای خود در نمی‌آید که با با خاصیت `clip` مشخص می‌شود.

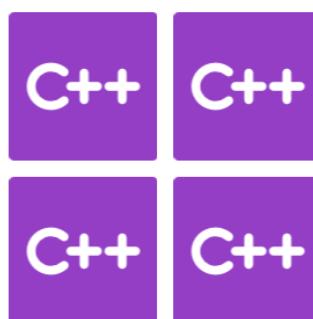
```
Flickable {  
    width: 200; height: 200  
    contentWidth: image.width; contentHeight: image.height  
    clip: true  
    Image { id: image; source: "qrc:/c-logo.png" }  
}
```

به کد فوق دقت کنید، جهت استفاده از `Flickable` و تاثیر آن بر روی آیتم یا شیء مورد نظر می‌باشد خاصیت `contentWidth` و `contentHeight` آن را برابر مقادیر مربوطه در شیء دیگر که به عنوان فرزند یا محتوا خواهد بود قرار دهیم. با فعال بودن خاصیت `clip` امکان نمایش محتوا در محدوده مشخص شده کنترل را فراهم می‌کند. در اینجا دامنه ۲۰۰ در ۲۰۰ مگاپیکسل است. برای اینکه این محدودیت از بین برود، می‌توان به روش اختصاصی اکتفا کرد و مقدار `clip` را با `true` مشخص کرد و یا اینکه توسط لنگر مربوطه محدوده آن را گسترش و یا مقادیر طول و عرض را افزایش داد.

تصویر زیر در حالت `clip` **فعال** است:



همچنین تصویر بعدی مربوط به وضعیت آن در حالت `clip` **غیرفعال** است:

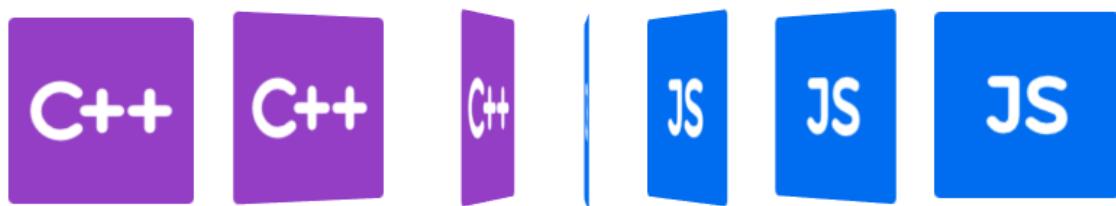


نکته ۱: در صورتی که کد مربوط به `clip` نوشته نشود خاصیت `clip` به صورت پیش فرض غیرفعال خواهد شد.

نکتهٔ ۲ : با توجه به جزئیات اجرایی، آیتم‌های قرار داده شده در داخل بیم **Flickable** نمی‌توانند توسط شناسه id به صورت لنگر - anchors به عنوان والد استفاده شوند.

## نوع **Flipable**

نوع **Flipable** آیتمی است که می‌تواند بین جلو و عقب به صورت جهش و چرخشی - برگردان نمایش داده شود. مانند یک کارت ویزیت که در دو طرف محتوای متفاوتی دارد. انواع و حالت‌های نمایشی آن نیز توسط State و Transition قابل تغییر است. خاصیت‌های عقب و جلو (Front و Back) برای نگه داری آیتم‌هایی استفاده می‌شوند که ترتیب نمایش عقب و جلو آیتم را مشخص می‌کنند.



```
Flipable {
    id: flipable
    width: 240
    height: 240

    property bool flipped: false
    front: Image { source: "qrc:/c-logo.png"; anchors.centerIn: parent }
    back: Image { source: "qrc:/js-file.png"; anchors.centerIn: parent }

    transform: Rotation {
        id: rotation
        origin.x: flipable.width/2
        origin.y: flipable.height/2
        axis.x: 0; axis.y: 1; axis.z: 0
    }

    states: State {
```

```

name: "back"

PropertyChanges { target: rotation; angle: 180 }

when: flipable.flipped

}

transitions: Transition {

    NumberAnimation {

        target: rotation; property: "angle"; duration: 2000

    }

}

MouseArea {

    anchors.fill: parent

    onClicked: flipable.flipped = !flipable.flipped

}

}

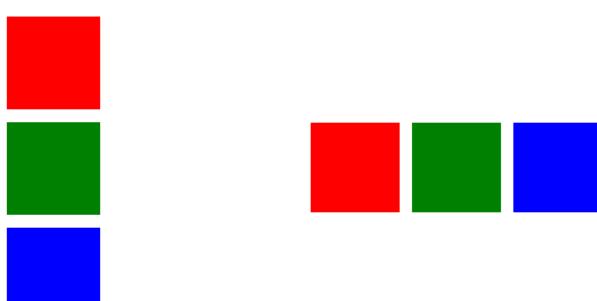
```

در نمونه کد بالا خاصیت `flipped` از نوع بولین برابر `false` قرار گرفته است این به این معنی است که هنگام اجرا به صورت خودکار وضعیت چرخش و تغییر صورت نخواهد گرفت. در خط بعد صفات `back` و `front` این کنترل برابر با دو کنترل تصویر است که هر یک دارای یک فایل تصویر متفاوت هستند. سپس خاصیت تغییر شکل شیء با نوع چرخشی `Rotation` برای مبداء های مربوطه در محورهای `x` و `y` و `z` تنظیم شده است.

در ادامه، وضعیت آن توسط `state` با نام `back` با ویژگی چرخشی `180` درجه مشخص شده است و سپس نحوه انتقال در زاویه بر تصویر یا سوی دیگر شیء توسط `Transition` مشخص شده است که در بازه زمانی `2000` میلی ثانیه - `duration` با مشخصه زاویه آن یعنی `angle` تغییر خواهد کرد. و در آخر در خاصیت کلیک شدن ماوس وضعیت روی برگردان شیء مشخص می شود که با یک بار کلیک روی آن و کلیک مجدد پشت شیء که هر یک دارای تصاویر مشخصی هستند نمایان خواهند شد.

## نوع Flow

آیتم Flow آیتمی است که موقعیت آیتم های فرزند را همانند کلمات موجود بر روی یک صفحه مشخص می کند. برای مثال بسته بندی آیتم ها در ردیف یا ستون را فراهم می کند.



استفاده از آن به شکل زیر است کافی است آیتم‌ها را در بدنه آن قرار داده و از خاصیت دسته بنده کننده آن استفاده کنید.

```
Flow {  
    anchors.fill: parent  
    anchors.margins: 4  
    spacing: 10  
    Rectangle {  
        color: 'red'  
        width: 72  
        height: 72  
    }  
    Rectangle {  
        color: 'green'  
        width: 72  
        height: 72  
    }  
    Rectangle {  
        color: 'blue'  
        width: 72  
        height: 72  
    }  
}
```

در کد ذکر شده سه شیء مستطیل با رنگ های قرمز، سبز و آبی با ابعاد ۷۲ در ۷۲ پیکسل معین شده‌اند که با قرار گیری در داخل آیتم Flow زمانی که پنجره مربوطه نسبت به ابعاد صفحه تغییر کند محتوای آن نیز بر اساس آن مرتب و پاسخپو خواهد شد. این خاصیت یکی از مواردی است که در پیاده سازی رابط کاربری Responsive بسیار کاربرد خواهد داشت.

## نوع FocusScope

آیتم FocusScope برای مدیریت کردن تمکز (فکوس) بر روی صفحه کلید مورد استفاده قرار می‌گیرد. این آیتم زمانی مورد استفاده قرار می‌گیرد که از یک کامپونت تحت QML برای استفاده‌های مجدد فراخوانی شود در این صورت توسط آن دامنه‌ی متمنکز بر صفحه کلید را می‌توان مشخص کرد.

```

FocusScope {
    id: scope
    focus: true
    width: rectangle.width; height: rectangle.height
    Rectangle {
        id: rectangle
        anchors.centerIn: parent
        color: "grey";
        width: 260; height: 30;
        Text { id: label; anchors.centerIn: parent }
        focus: true
        Keys.onPressed: {
            if (event.key === Qt.Key_A)
                label.text = 'Key A was pressed'
            else if (event.key === Qt.Key_B)
                label.text = 'Key B was pressed'
            else if (event.key === Qt.Key_C)
                label.text = 'Key C was pressed'
            else if (event.key === Qt.Key_d)
                label.text = 'Key d was pressed'
        }
    }
}

```

در کد فوق با قرار دادن شیء **FocusScope** در داخل **Rectangle** و استفاده از رویدادهای **Keys** در نظر داریم با فشرده شدن کلیدهای A, B, C, D وضعیت عملی را انجام دهیم. در حالت عادی دسترسی به خاصیت تمرکز بر روی صفحه کلید غیرفعال است بنابر این برای دسترسی به آن میبایست در دامنهاین کنترل خاصیت **focus** را برابر **true** قرار دهیم. این خاصیت در فرم‌های تودرتو بسیار پر کاربرد است لذا زمانی که شما نیاز دارید در داخل یک فرم دیگر فرمی را مورد استفاده قرار دهید که نیاز است مقادیری توسط صفحه کلید در آن وارد شود میبایست از این خاصیت استفاده کنید.

## نوع **FontLoader**

نوع **FontLoader** برای بارگذاری فونت توسط نام و یا آدرس خاص است.

نوع	صفت
string	<b>name</b>
url	<b>source</b>
enumeration	<b>status</b>

ساختار کلی آن به صورت زیر است :

```
FontLoader {
    id: webFont;
    source: "http://www.mysite.com/myfont.ttf"
}
```

به طور کلی این کنترل با داشتن خاصیت name به صورت رشتہ نام - یا همان مشخصه از خانواده فونت را نگه می‌دارد. برای مثال فونت Tahoma با مشخصه فراخوانی می‌شود.

جهت استفاده از آن در کنترل‌های مورد نظر می‌باشد خاصیت font.family کنترل مربوطه را برابر با نام فونت قرار داد.  
در مثال زیر روش آن مشخص گردیده است:

```
Item {
    width: 200; height: 50

    FontLoader {
        id: webFont
        source: "http://font.genyleap.com/tahoma.ttf" }

    Text {
        text: "My Font"
        font.family: webFont.name
    }
}
```

با فراخوانی نوع FontLoader در آیتم font.family مورد نظر خاصیت قرار می‌دهیم که توسط نوع بارگزار کننده فونت ایجاد شده است. در این صورت متن مربوطه بر اساس فونتی نمایان خواهد شد که مشخص شده است.

همچنین خاصیت source قابلیت پذیرش آدرس از روی وب سرور و همچنین سورس خود نرمافزار را فراهم میکند. برای مثال هر دو روش زیر

صحیح خواهد بود:

```
FontLoader { id: webFont; source: "http://font.genyleap.com/tahoma.ttf" }  
FontLoader { id: webFont; source: "qrc:/fonts/tahoma.ttf" }
```

خاصیت دیگری که این کنترل دارا است وجود وضعیت آن است. این خاصیت با نگه داری وضعیت بارگذاری فایل مربوط به قلم (فونت) نتیجه مرتبط با وضعیت آن را بر میگرداند که شامل وضعیت‌های زیر است.

نوع	صفت
این مقدار مشخص میکند که هیچ قلمی (فونتی) مشخص نشده است.	Null
این مقدار مشخص میکند که قلم (فونت) مورد نظر با موفقیت بارگذاری شده است.	Ready
این مقدار مشخص میکند که قلم (فونت) مورد نظر در حال بارگذاری است.	Loading
این مقدار مشخص میکند که موقع بارگذاری قلم (فونت) مشکلی رخ داده است.	Error

هنگام بارگذاری موفقیت آمیز فایل مربوط به قلم (فونت) پیغامی با عنوان "Loaded" را در کنسول چاپ خواهد شد.

```
FontLoader {  
    id: loader  
    onStatusChanged: {  
        if (loader.status == FontLoader.Ready) {  
            console.log('Loaded')  
        }  
    }  
}
```

## FontMetrics نوع

نوع اندازه کاراکترها و رشته‌هایی که برای فونت مشخص شده است را محاسبه میکند. این نوع زیر مجموعه‌ای از رابطه‌های برنامه‌نویسی C++ را فراهم میکند که با افزوده شدن آن قابلیت تغییر فونت با استفاده از ویژگی فونت محاسبه میشود. کد زیر نمونه مثالی از نحوه‌ی استفاده از این نوع است. همانطور که مشخص است اندازه شیء Rectangle بر اساس فونت بررسی و پیاده سازی میشود و بستگی به اندازه فونتی دارد که از قبل مشخص کرده‌اید. این روش بیشتر در بهینه سازی طراحی متن و هماهنگی آن با اشیاء مورد استفاده قرار میگیرد.

```

FontMetrics {
    id: fontMetrics
    font.family: "Arial"
}

Rectangle {
    width: fontMetrics.height * 6
    height: fontMetrics.height * 6
    color: "green"
}

```

## نوع Gradient

نوع (گرادیان) توسط دو یا چند رنگ ایجاد می‌شود، که در کنار هم یک رنگ یکپارچه و به هم آمیخته شده را تولید می‌کند. مجموعه رنگ‌ها توسط **GradientStop** به عنوان زیر والدی از آن، از بازه ۰.۰ تا ۱.۰ مشخص می‌شود. درصورتی که هیچ مشخصه‌ای از تعریف نشود نتیجه نهایی تولید شده به صورت یک رنگ جامد (تک رنگ) خواهد بود.

کد فوق ظاهر یک شیء **Rectangle** را توسط خاصیت آن **gradient** تغییر داده است که در سه مرحله خصیصه‌ی **GradientStop** تکرار شده است مقادیر موقعیت (position) و رنگ (color) وارد شده تا در نهایت ترکیب رنگ نارنجی، زرود و قرمز را با درصد تنظیم شده نمایان سازد.

```
Rectangle {
```

```
    width: 100; height: 100
```

```
    gradient: Gradient {
```

```
        GradientStop { position: 0.0; color: "Orange" }
```

```
        GradientStop { position: 0.50; color: "Yellow" }
```

```
        GradientStop { position: 1.0; color: "Red" }
```

```
}
```

```
}
```



نوع Grid با نام شبکه توری آیتم‌های فرزند را در داخل یک قاب توری (شبکه) ای قرار می‌دهد. یک Grid شبکه‌ای از سلول‌ها را می‌سازد که به اندازه کافی می‌تواند برای جای دادن تمامی آیتم‌ها مناسب و کافی باشد همچنین مکان آیتم در سلول‌ها را از چپ به راست و بالا به پایین تنظیم می‌کند. هر یک از آیتم‌ها در گوشش بالا سمت چپ قرار می‌گیرند.

یک کنترل توری (Grid) به طور پیش فرض چهار ستون را دربر می‌گیرد که برای تغییر و سفارشی سازی آن می‌بایست از بخش گزینه‌های مربوط به خاصیت آن اقدام کنید.

برای مثال، در زیر نوع توری که شامل چهار شیء مستطیل با اندازه‌های مختلف است را نشان می‌دهد.

```
Grid {
  columns: 2
  spacing: 5
  Rectangle { color: "red"; width: 100; height: 50 }
  Rectangle { color: "green"; width: 100; height: 50 }
  Rectangle { color: "blue"; width: 100; height: 85 }
  Rectangle { color: "orange"; width: 100; height: 48 }
}
```



دقت داشته باشید که اندازه بین سلول‌ها و اشیاء موجود در آن‌ها توسط خاصیت spacing مشخص می‌شود. تعداد ستون‌ها توسط columns و ردیف‌ها توسط مشخصه row مشخص می‌شوند.

نوع	صفت
این مقدار مشخص می‌کند که هیچ قلمی (فونتی) مشخص نشده است.	Null
این مقدار مشخص می‌کند که قلم (فونت) مورد نظر با موفقیت بارگذاری شده است.	Ready
این مقدار مشخص می‌کند که قلم (فونت) مورد نظر در حال بارگذاری است.	Loading
این مقدار مشخص می‌کند که موقع بارگذاری قلم (فونت) مشکلی رخداده است.	Error

شبکهٔ توری (GridMesh) نوعی مستطیل شکل متشکل از رؤوسی است که به طور برابر در روی شبکه تعریف شده‌اند که از آن برای تولید هندسه استفاده می‌شود. وضوح تور (شبکه) با خاصیت resolution با نوع عدد صحیح قابل تعریف است.

```
import QtQuick 2.9
import QtQuick.Window 2.2

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    ShaderEffect {
        width: 200
        height: 200
        mesh: GridMesh {
            resolution: Qt.size(20, 20)
        }
        property variant source: Image {
            source: "qrc:/logo.png"
            sourceSize { width: 200; height: 200 }
        }
        vertexShader: "
            uniform highp mat4 qt_Matrix;
            attribute highp vec4 qt_Vertex;
            attribute highp vec2 qt_MultiTexCoord0;
            varying highp vec2 qt_TexCoord0;
            uniform highp float width;
            void main() {
                highp vec4 pos = qt_Vertex;
                highp float d = .5 * smoothstep(0., 1., qt_MultiTexCoord0.y);
                pos.x = width * mix(d, 1.0 - d, qt_MultiTexCoord0.x);
                gl_Position = qt_Matrix * pos;
            }
        "
    }
}
```

```

    qt_TexCoord0 = qt_MultiTexCoord0;
}

}

}

```

## GridView نوع

کنترل GridView داده‌هایی را از مدل QML مانند XmlListModel و ListModel یا کلاس‌های سفارشی شده از طرف C++ را که از طرف کلاس QAbstractListModel ارث بری شده است تعریف می‌کند. به طور کلی یک گرید ویو دارای قابلیت مدلینگ است که داده‌ها را برای نمایش تعریف می‌کند. همچنین مشخص می‌کند که نمایش آن باید چگونه باشد. آیتم‌ها در این نوع به صورت افقی و یا عمودی نمایش داده می‌شوند. کد مربوط به گرید ویو به صورت زیر است :

```

GridView {
    id: grid

    anchors.fill: parent
    anchors.margins: 10
    anchors.centerIn: parent

    cellWidth: 80; cellHeight: 80
    model: ContactModel {}

    delegate: contactDelegate

    highlight: Rectangle {
        color: "#ccc"; radius: 10;
    }

    focus: true
}

```

در این بخش مشخصه‌های cellHeight و cellWidth فاصله مربوط به سلول‌های نمایش دهنده را مشخص کرده است سپس ترجیحاً توسط یک کامپوننت از قبل تعریف شده در یک فایل با نام ContactModel.qml داده‌های مربوطه را از طریق مشخصه model دریافت کرده است که داده‌ها در ادامه به صورت زیر در فایل مربوطه آورده شده‌اند.

```

ListModel {
    ListElement {
        name: "User A001"
        userIcon: "qrc:/user.png"
    }
}

```

```

ListElement {
    name: "User A002"
    userIcon: "qrc:/user.png"
}

ListElement {
    name: "User A003"
    userIcon: "qrc:/user.png"
}

ListElement {
    name: "User A004"
    userIcon: "qrc:/user.png"
}

}

```

در این بخش مدلی از لیست داده‌ای ایجاد شده است که توسط نوع `ListModel` نگه داری می‌شوند. داده‌ها بر اساس المان‌های `name` و `userIcon` مشخص می‌شوند. مشخصه اول رشته مرتبط با نام را دریافت کرده و مشخصه دوم مسیر تصویر مربوط به ایکون مربوطه را دریافت می‌کند. در ادامه با مشخصه `delegate` در گرید ویو آیتم‌ها به صورت سفارشی شده در داخل یک کامپوننت وارد شده‌اند که به صورت زیر است :

```

Component {
    id: contactDelegate

    Item {
        width: grid.cellWidth; height: grid.cellHeight

        Column {
            anchors.fill: parent
            Image { width: 64; height: 64;
                    sourceSize.width: 64; sourceSize.height: 64;
                    source: userIcon; anchors.horizontalCenter: parent.
horizontalCenter;
            }
            Text { text: name;
                    anchors.horizontalCenter: parent.horizontalCenter
            }
        }
    }
}

```

```
}
```

```
}
```

در کد فوق با استفاده از کلمه کلیدی Component آیتمی را به صورت کامپوننت برای مشخصه delegate در گردید و بتوان در نظر گرفته ایم. در این بخش توجه داشته باشید که برای شناسایی داده ها می بایست از مشخصه های تعریف شده به صورت سفارشی استفاده شود. برای مثال جهت فراخوانی نام از مشخصه name و تصویر مربوطه از مشخصه userIcon استفاده خواهد شد. که به ترتیب برابر با سورس تصویر و مقدار متن در نظر گرفته شده اند.



## نوع Image

نوع Image وظیفه نمایش یک تصویر را بر عهده دارد. مسیر منبع آن توسط آدرس URL توسط مشخصه source مشخص می شود. همچنین قالب هایی که این کنترل پشتیبانی می کند عبارتند از : SVG, JPEG, PNG و

ساختار و نحوه استفاده از این کنترل به صورت زیر است :

### Image {

```
source: "qrc:/logo.png"
```

```
}
```

نوع	صفت
bool	<b>asynchronous</b>
bool	<b>autoTransform</b>
bool	<b>cache</b>
enumeration	<b>fillMode</b>
enumeration	<b>horizontalAlignment</b>
bool	<b>ipmap</b>
bool	<b>mirror</b>
real	<b>paintedHeight</b>
real	<b>paintedWidth</b>
real	<b>progress</b>
bool	<b>smooth</b>
url	<b>source</b>
QSize	<b>sourceSize</b>

enumeration	<b>status</b>
enumeration	<b>verticalAlignment</b>

توسط خاصیت `fillMode` می‌توان نحوه نمایش تصویر را مشخص کرد. در نمونه زیر برای (حفظ تناسب ابعاد) از مشخصه `Image.PreserveAspectFit` استفاده شده است.

```
Image {
    width: 150
    height: 150
    source: "qrc:/logo.png"
    fillMode: Image.PreserveAspectFit
}
```



در حالت بعدی برای اصلاح و برش اضافات تصویر از مشخصه `verticalAlignment` و `horizontalAlignment` استفاده می‌شود که در نتیجه آن تصویر بر اساس ابعاد درج شده برش یافته و نمایان خواهد شد که با مشخصه `Image` و گزینه‌های مشابه موجود در CSS3 تنظیم می‌شود.

```
Image {
    source: "qrc:/logo.png"
    width: 150; height: 150
    fillMode: Image.Tile
    horizontalAlignment: Image.AlignLeft
    verticalAlignment: Image.AlignTop
}
```



خاصیت `sourceSize` با مشخصه‌های `width` و `height` نگه دارنده اندازه واقعی سورس تصویر بارگیری شده است. برخلاف خاصیت‌های `width` و `height` در `sourceSize` مقایس تصویر را در حافظه مشخص می‌کند. و این زمانی مورد استفاده قرار می‌گیرد که نیازی نیست از حداقل اندازه تصویر در حافظه استفاده شود.

```
Image {
    sourceSize.width: 150
    sourceSize.height: 150
    source: "qrc:/checked.png"
}
```



در مثال مذکور فایل checked.png از اندازه ۲۵۶ پیکسلی برخوردار است که آن را بدون دستکاری توسط مشخصه sourceSize به ۱۵۰ در ۱۵۰ پیکسل کاهش داده‌ایم.

کنترل Image یک خاصیتی با نام status دارد که در آن وضعیت بارگیری فایل تصویر مشخص می‌شود که شامل چهار مشخصه Null, Ready, Loading و Error است. در صورتی که سورس مربوطه دارای تصویر فایل نباشد مقدار بازگشتی آن Null خواهد بود. اگر فایل مربوطه با موفقیت بارگیری شده باشد مقدار بازگشتی آن Ready خواهد بود. مقدار بازگشتی مربوط به زمان در حال بارگیری Loading و در نهایت در صورت بروز هرگونه خطایی در بارگیری تصویر Error بازگشت داده خواهد شد.

جهت برنامه ریزی برای تغییر وضعیت و اعلام وضعیت مرتبط با نوع تصویر از مشخصه رویداد onStatusChanged استفاده می‌کنیم. کد آن به صورت زیر خواهد بود:

```
onStatusChanged: {
    if (image.status == Image.Ready)
        statusText.text = "Okey!";
        statusText.color = "green"

    }
else if (image.status == Image.Loading) {
    statusText.text = "Wait...";
    statusText.color = "orange"
}

else if (image.status == Image.Error) {
    statusText.text = "Error!";
    statusText.color = "red"
}

else if (image.status == Image.Null) {
    statusText.text = "No image found!";
    statusText.color = "yellow"
}
}
```

در کد فوق توسط دستور شرطی if تحت شناسه تصویر با نام image وضعیت آن را برابر با رخدادهای احتمالی تعریف کردہایم. سپس با توجه به هر یک از حالت‌ها مشخصه statusText که مربوط به یک نوع رشته‌ای برای نمایش متن و رنگ آن است اختصاص دادهایم که به صورت زیر تکمیل می‌شود.

```
Text { id: statusText }
```

خروجی مربوط به دستور شرطی بر اساس وضعیت نهایی به صورت زیر خواهد بود.



Okey!

روش نگه دارنده (Trigger) با تغییر حالت (State) به صورت زیر است :

```
State { name: 'loaded'; when: image.status == Image.Ready }
```

جهت سفارشی سازی بر اساس State (وضعیت) مربوطه به صورت زیر کد آن را می‌نویسیم:

```
states: [
    State { name: 'loaded';
        when: image.status == Image.Ready
        PropertyChanges { target: statusText; text: "OK!" }
    },
    State { name: 'error';
        when: image.status == Image.Error
        PropertyChanges { target: statusText; text:"Error!" }
    }
]
```

خاصیت حالت یا همان State در صفحات آتی کتاب توضیح داده شده است که می‌توانید از طریق مستندات مرتبط با این خاصیت برای تمامی انواع QML استفاده کنید.

همچنین روش سومی هم برای استفاده از وضعیت نوع با عنوان بایندیگ (Binding) یا همان اتصال موجود است که در کنترل مقصد یا نمایش دهنده نتیجه از آن استفاده خواهد شد.

```
Text { text: image.status == Image.Ready ? 'Loaded' : 'Not loaded' }
```

## نوع IntValidator

نوع IntValidator عمل اعتبار سنجی برای مقادیر صحیح را ارائه می‌کند. معمولاً برای اعتبار سنجی کنترل‌های ورودی مانند TextField مورد استفاده قرار می‌گیرد و شکل اصلی آن به صورت زیر است:

```
IntValidator { }
```

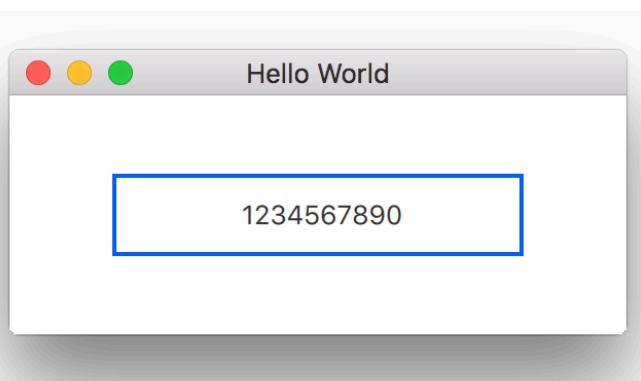
نوع	صفت
int	bottom
string	locale
int	top

نمونه مثال آن به صورت زیر خواهد بود که با برابر بودن با مشخصه validator کنترل مورد نظر از آن استفاده می‌شود.

```
TextField {  
    anchors.centerIn: parent  
    placeholderText: "0"  
    validator: IntValidator{}  
    horizontalAlignment: TextInput.AlignHCenter  
    onTextChanged: { console.log(parseInt(text,10))}  
}
```

در صورتی که مقدار صحیحی وارد شود توسط IntValidator مقادیر ارزیابی شده و چاپ خواهند شد.

```
qml: 1  
qml: 12  
qml: 123  
qml: 1234  
qml: 12345  
qml: 123456  
qml: 1234567  
qml: 12345678  
qml: 123456789  
qml: 1234567890
```



خاصیت bottom حداقل مقدار ممکن را تعریف می‌کند که به صورت پیش فرض برابر است با **-۲۱۴۷۴۸۳۶۴۷** و خاصیت top حداقل تعداد مقادیر ممکن را می‌پذیرد که به صورت پیشفرض مقدار آن برابر است با **۲۱۴۷۴۸۳۶۴۷** که می‌توانید آن را سفارشی سازی نمایید.

```
validator: IntValidator{ bottom:0; top:9999999999; }
```

همچنین خاصیت locale نگه دارنده‌ای برای تفسیر مقادیر بر() است.

## Item نوع

نوع Item یک نوع برای تمامی آیتم‌های بصری در Qt Quick است. تمامی آیتم‌های بصری در کیوت کوئیک از نوع Item ارث بری می‌کنند. اگرچه نوع Item هیچ جلوه ظاهری را ندارد اما تمامی خاصیت‌ها و ویژگی‌های سراسری اقلام بصری را مانند مختصات x و y و یا width و height را تعریف می‌کند.

نوع Item می‌تواند برای گروه بندی چندین آیتم متفاوت در یک سند ریشه‌ای مفید باشد که نحوه استفاده از آن در کد زیر آورده شده است :

```
import QtQuick 2.9

Item {
    Rectangle {
        color: "red"
        width: 64
        height: 64
    }

    Text {
        text: "Hello World!";
    }

    Image {
        source: "qrc:/logo.png"
    }
}
```

## برخی از ویژگی‌ها مانند برقراری ارتباط با کلیدها:

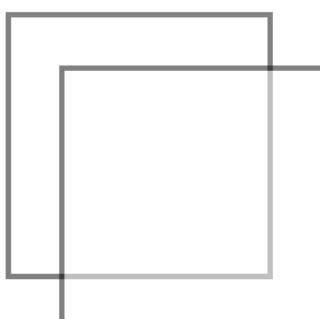
دست زدن به کلیدها تحت انواع بصری مبتنی بر آیتم میسر است. سیگنال‌های موجود در کلیدها مانند فشرده شدن یا رها شدن آنها یا onReturnPressed سیگنال‌های مشخص دیگری مانند کلیدهای سفارشی شده تحت نوع آیتم مورد کنترل قرار می‌گیرند. که تحت رویدادی با نام ارسال می‌شوند.

```
Item {  
    focus: true  
  
    Keys.onPressed: {  
  
        if (event.key === Qt.Key_Up) {  
            console.log("Move Up");  
            event.accepted = true;  
        }  
    }  
  
    Keys.onReturnPressed: console.log("Pressed return");  
}
```

در کد فوق مشخصه focus عمل فکوس شدن بر روی آیتم را انجام می‌دهد. توسط رویداد onPressed تحت نگه دارنده Keys می‌توان نسبت به رخدادهای مرتبط با فشرده شدن کلیدها دستورات سفارشی را فراهم آورد. در کد مذکور هنگام فشرده شدن کلید جهت رو به بالا در کنسول محیط توسعه متن Move Up نمایان خواهد شد.

در نهایت ویژگی طرح معکوس یکی دیگر از ویژگی‌های نوع آیتم است که این امکان را فراهم می‌سازد تا هر نوع طرح بصری بر روی این آیتم به صورت معکوس نمایان گردد و این زمانی است که از مشخصه LayoutMirroring در آیتم مربوطه استفاده گردد. یکی از نکاتی که باید به آن توجه کرد این است که در نوع Item ویژگی ظاهری وجود ندارد. بنابراین در صورتی که از آن به عنوان یک لایه استفاده می‌کنید فاقد رنگ یا نمایه خاص خواهد بود. برای مثال در کد زیر در داخل یک آیتم از نوع مستطیل استفاده شده است.

```
Item {  
    id: nonLayered  
  
    anchors.centerIn: parent  
  
    opacity: 0.5  
  
    width: 100  
  
    height: 100  
  
    Rectangle {
```



```

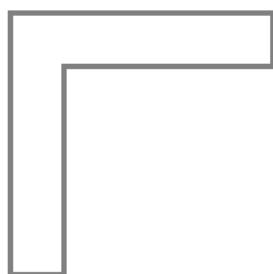
width: 100;
height: 100;
border.width: 2
}

Rectangle {
  x: 20;
  y: 20;
  width: 100;
  height: 100;
  border.width: 2
}
}

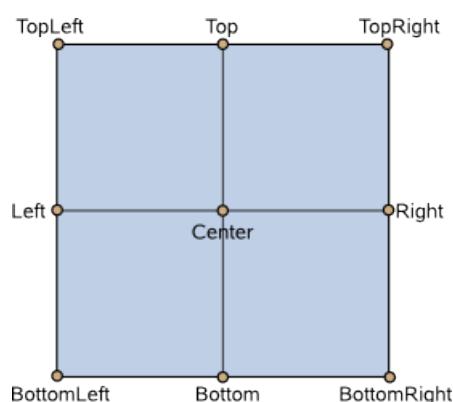
```

حال در صورتی که خاصیت لایه آیتم را فعال نماییم به صورت زیر لایه آن نیز بخشی از آیتم‌های بعدی در نظر گرفته خواهد شد.

```
layer.enabled: true
```



ویژگی transformOrigin نقاط مبدأ که در اطراف مقیاس و چرخش واقع شده است را نگه می‌دارد. به تعداد ۹ قسمت از این نقاط در دسترس هستند طبق تصویر زیر که نشان داده می‌شود منشاء به طور پیشفرض مرکز آیتم است.



```

Image {
    source: "qrc:/logo.png"
    transformOrigin: Item.BottomRight
    rotation: 45
}

```

کد فوق توسط مشخصه `transformOrigin` از نقطه سمت راست آیتم آن را به مقدار ۴۵ درجه چرخش می‌دهد که نتیجه آن به صورت زیر خواهد بود.



## نوع `ItemGrabResult`

نوع `ItemGrabResult` به عنوان یک ظرف کوچک برای کپسوله‌سازی نتایج حاصل از آیتم توسط `Item::grabToImage()` است. به طور کلی برای به دست آوردن یک آیتم به صورت یک تصویر در حافظه مورد استفاده قرار می‌گیرد. گرفتن نتیجه به طور غیر همزمان صورت می‌گیرد و تابع `javaAsseripit` آن را زمانی که تکمیل می‌شود فراهم می‌کند. برای مثال تبدیل بخشی از آیتم‌ها بر روی فرم به صورت فایل تصویری.

```

var stat = source.grabToImage(function(result) {
    result.saveToFile("yourfile.png"); //saves to a file
});

```

کد زیر تصویر مربوط به‌ایتم مورد نظر را دریافت و در حافظه نگه داری می‌کند، سپس آن را در سورس تصویر نمایش می‌دهد.

```

source.grabToImage(function(result) {
    image.source = result.url;
},
Qt.size(50, 50));

```

در کد تصویر در ابعاد ۵۰ در ۵۰ پیکسل از آیتم را در نوع تصویر `Image` قرار می‌دهد. این روش برای دریافت اسکرین شات و یا ثبت بخشی از فرم یا فایل کاربرد بسیاری دارد. لذا جهت توسعه آن می‌بایست در بک اند برنامه کد مبدل و ذخیره سازی آن در C++ ارائه شود که به صورت زیر خواهد بود.

ابتدا کلاسی با نام Generator ساخته و در فایل generator.h کد زیر را پیاده سازی خواهیم کرد:

```
#ifndef GENERATOR_H
#define GENERATOR_H

#include <QVariant>
#include <QObject>
class generator : public QObject
{
    Q_OBJECT
public:
    generator();
    Q_INVOKABLE void takeShot(QVariant data);
};

#endif // GENERATOR_H
```

در کد مذکور کلاس مربوط به سازنده ساخته شده و تحت کلمه کلیدی **Q\_INVOKABLE** تابعی با ورودی QVariant تعریف شده است که دریافت کننده نوع تصویر از طرف آیتم QML خواهد بود.

سپس در فایل generator.cpp کد مربوطه به صورت زیر خواهد بود :

```
#include "generator.h"
#include <QPrinter>
#include <QPainter>
#include <QPrintDialog>

#include <QPixmap>
#include <QImage>
generator::generator()
{
}

void generator::takeShot(QVariant data)
{
    QImage img = qvariant_cast<QImage>(data);
```

```

QPrinter printer;
QPrintDialog *dlg = new QPrintDialog(&printer, 0);
if(dlg->exec() == QDialog::Accepted) {
    QPainter painter(&printer);
    painter.drawImage(QPoint(0,0), img);
    painter.end();
}

```

تابع takeShot توسط کلاس QPrinter و QImage پیاده سازی شد است سپس پنجره چاپ و ذخیره سازی توسط آن نمایش و فایل ارسالی قابل ذخیره خواهد بود. که توسط QML دریافت و اجرا خواهد شد. کافی است تابع takeShot را از طرف C++ در QML رجیستر کرده و آن را فراخوانی کنید (توجه داشته باشید که مازول printsupport را در فایل pro. تعريف کرده باشید).

## نوع KeyEvent

همانطور که می دانید یکی از مهمترین بخش های کد نویسی مرتبط با انواع حالت ها و رخدادها بر روی صفحه کلید است. نوع KeyEvent اطلاعاتی در رابطه با کلیدها را فراهم می کند. ساختار و نحوه استفاده از آن به صورت زیر است :

```

Item {
    focus: true
    Keys.onPressed: {
        if (event.key === Qt.Key_Space)
            console.log("Key Space was pressed!");
    }
}

```

در کد فوق هنگام فشرده شدن کلید فاصله (Space) پیغام مربوط به فشرده شدن ان در کنسول چاپ خواهد شد.

نوع	صفت
bool	accept
int	count
bool	isAutoRepeat
int	key
int	modifiers
qunit32	nativeScanCode

string	text
--------	------

- صفت accept در صورتی که فعال باشد، تمامی رویدادهای برگرفته شده از آیتم‌های والد آن مسدود خواهد شد. در واقع با فعال بودن این ویژگی عمل جلوگیری از رویدادهای کلیدی صورت خواهد گرفت.
- صفت count تعداد کلیدهای درگیر شده با رویدادها را نگه می‌دارد. مشخصه isAutoRepeat با نوع بولین مشخص می‌کند که‌ایا رویداد مربوطه از طرف کلید به صورت خود کار منتشر شده است یا خیر.
- همچنین صفت key کد مربوط به فشرده شدن و رها شدن کلید را در خود نگه می‌دارد که توسط مشخصه Qt.Key می‌توان به لیست کلیدهای موجود دسترسی پیدا کرد. در صورتی که مقدار آن برابر ۰ یا Qt.Key\_Unknown باشد نشانگر این است که هیچ نتیجه‌ای از هیچ یک از کلیدها دریافت نشده است.
- صفت modifiers نگه دارنده انواع پرچم (flag)‌ها برای کلیدهای ترکیبی است که شامل Qt.NoModifier به معنی فشرده شدن بدون Qt.ControlModifier به عنوان فشرده شدن کلید Shift است. مشخصه Qt.ShiftModifier دهنده یا ترکیبی است. به عنوان فشرده شدن کلید Ctrl بوده و کلیدهای بعدی مانند Qt.AltModifier برای کلید Alt و مشخصه Qt.MetaModifier برای کلید Meta کد زیر مثالی از فشرده شدن کلیدهای ترکیبی Shift و Space است :

```
Item {
    focus: true

    Keys.onPressed: {
        if ((event.key === Qt.Key_Space) &&
            (event.modifiers & Qt.ShiftModifier))
            console.log("Keys Shift + Space was pressed!");
    }
}
```

## نوع KeyNavigation

نوع KeyNavigation وظیفه پشتیبانی از کلیدهای جهت دار برای ناوبری را بر عهده دارند. معمولاً رابط‌کاربری مبتنی بر کلید اجازه استفاده از کلیدهای جهت دار را برای ناوبری - حرکت بین آیتم‌های مرکز (فکوس) شده را فراهم می‌کند. کد زیر نشانگر نحوه استفاده از نوع KeyNavigation در بین آیتم‌های یک توری ۲ در ۲ را نمایش می‌دهد.

```
Grid {
    width: 500; height: 500
```

```

columns: 2

Rectangle {
    id: topLeft
    width: 50; height: 50
    color: focus ? "red" : "lightgray"
    focus: true
    KeyNavigation.right: topRight
    KeyNavigation.down: bottomLeft
}

Rectangle {
    id: topRight
    width: 50; height: 50
    color: focus ? "red" : "lightgray"
    KeyNavigation.left: topLeft
    KeyNavigation.down: bottomRight
}

Rectangle {
    id: bottomLeft
    width: 50; height: 50
    color: focus ? "red" : "lightgray"
    KeyNavigation.right: bottomRight
    KeyNavigation.up: topLeft
}

Rectangle {
    id: bottomRight
    width: 50; height: 50
    color: focus ? "red" : "lightgray"
    KeyNavigation.left: bottomLeft
    KeyNavigation.up: topRight
}

```



در کد فوق مشخصه KeyNavigation با مقدار گیری چهار جهت (کلیدهای جهت دار) به شناسه آیتمهای قبل و بعد از خود متصل می‌شوند. و با فشرده شدن کلیدهای جهت دار با متمرکز شدن بر روی آیتمها رنگ آنها تغییر و به حالت انتخاب تغییر می‌یابد.

## نوع Keys

نوع Keys لیستی از کلیدها را برای فشرده شدن فراهم می‌کند.



تمامی اشیاء هندسی بصری از خاصیت‌های فشرده شدن توسط کلید را پشتیبانی می‌کنند. کلیدها می‌توانند از طریق سیگنال‌های onPressed و onReleased مدیریت شوند.

بنابراین امکان قابلیت لمسی بر روی اشیاء توسط نوع Keys فراهم می‌شود. در قطه کد زیر با فشرده شدن کلید جهت راست متن مربوط به آن بر روی صفحه کنسول نمایش داده می‌شود که نشانگر دریافت پاسخ از طرف کلید فشرده شده است.

```
Item {  
    anchors.fill: parent  
    focus: true  
  
    Keys.onPressed: {  
        if (event.key === Qt.Key_Right) {  
            console.log("Move Right");  
            event.accepted = true;  
        }  
    }  
}
```

نوع LayoutMirroring مشخصه‌ای است که قابلیت انعکاس بر عکس به صورت آینه را فراهم می‌کند. کاربرد بسیاری در راست چین و چپ چین کردن طراحی دارد. یک ویژگی متصل برای استفاده از حالت آینه‌ای آیتم‌ها و حالت‌های مختلف آن مورد استفاده قرار می‌گیرد که شامل آیتم‌های Row و Grid و نمایه‌هایی همچون ListView و GridView افقی است. حالت معکوس کننده‌ک تغییر بصری است که از طریق لنگر گاه‌های چپ و راست اعمال می‌شود. همچنین آیتم‌های فرزند را نیز تحت تاثیر قرار می‌دهد.

معکوس شدن یک آیتم با فعال بودن خاصیت enabled امکان پذیر است که مقدار آن برابر true خواهد بود. به صورت پیشفرض تاثیر آن بر روی خود آیتم بوده و در صورتی که نیاز باشد آیتم‌های فرزند هم تحت تاثیر قرار گیرند خاصیت childrenInherit را برابر true قرار خواهیم داد.

**توجه:** در نسخه ۵.۸ به بعد کیوت مشخصه LayoutMirroring را می‌تواند توسط Window بکار برد.

```
Rectangle {
    LayoutMirroring.enabled: true // فعال‌سازی معکوس
    LayoutMirroring.childrenInherit: true // تاثیر بر روی فرزند
    width: 300; height: 50
    color: "green"
    border.width: 1
}
Row {
    anchors { left: parent.left; margins: 5 }
    y: 5; spacing: 5
    Repeater {
        model: 5
        Rectangle {
            color: "white"
            opacity: (5 - index) / 5
            width: 40; height: 40
            Text {
                text: index + 1
                anchors.centerIn: parent
            }
        }
    }
}
```

```

    }
}

}

```



در کد زیر در رابطه با قانون راست چین در زبان فارسی مثالی آورده‌ایم که توسط خاصیت LayoutMirroring به راحتی راست به چپ و چپ به راست را تنظیم می‌کند.

```

Rectangle {
    LayoutMirroring.enabled: true
    LayoutMirroring.childrenInherit: true
    Row {
        spacing: 10
        Text {
            id: name
            text: qsTr("نام و نام خانوادگی")
        }
        Text {
            text: "کامبیز اسدزاده"
        }
    }
}

```

Firstname & Lastname: Kambiz Asadzadeh  
نام و نام خانوادگی: کامبیز اسدزاده

## ListView نوع

نوع ListView یک لیست از آیتم‌ها را فراهم می‌کند. یک ListView به طور کلی داده‌هایی را از طریق مدل‌های ساخته شده نمایش می‌دهد که از انواع QML تشکیل شده‌اند مانند :ListModel و XmlListModel و یا کلاس‌های سفارشی شده از طرف C++ که توسط کلاس‌های QAbstractListModel و QAbstractItemModel فراهم می‌شوند.

یک کنترل لیست دارای مدل بوده و داده‌های تعریف شده در آن به نمایش در می‌آیند به عنوان محول کننده - نماینده (delegate) نحوه نمایش آنها سفارشی سازی می‌شود و آیتم‌ها در این کنترل به دو حالت عمودی و افقی قرار می‌گیرند و نحوه نمایش آنها به صورت لغزنده از طریق وراشت Flickable نمایان می‌شود.

ساختار و نحوه استفاده از آن به صورت زیر است :

```
ListView {  
    model: ContactModel {}  
    delegate: contactDelegate  
    highlight: Rectangle { color: "#f1f1f1"; radius: 5 }  
    focus: true  
    orientation: ListView.Horizontal  
}
```

با تعریف ListView خاصیت model وظیفه نگه داری داده‌های مدلی را در اختیار دارد که توسط ListModel و عناصر آن معین می‌شود که به صورت زیر در یک فایل جداگانه آن را تعریف کرده‌ایم :

```
import QtQuick 2.9  
  
ListModel {  
    ListElement {  
        name: "Kambiz Asadzadeh"  
        number: "044 44632"  
        colorCode : "red"  
    }  
    ListElement {  
        name: "Araz Ostadi"  
        number: "044 54332"  
        colorCode : "blue"  
    }  
    ListElement {  
        name: "Amin Ghasemi"  
        number: "044 74232"  
        colorCode : "green"  
    }  
}
```

```
    }  
}
```

وظیفه ListModel در صفحات بعدی کتاب معرفی شده است در این بخش باید در نظر داشته باشیم این مشخصه امکان تعریف آیتم و تشکیل یک مدل داده‌ای را برای ارسال به مشخصه model در ListView را فراهم می‌کند که به صورت جدا گانه در یک فایل با نام ContactModel.qml نوشته شده است.

مشخصه‌های نام - شماره و کد رنگ به ترتیب با شناسه‌های name, number و colorCode تعریف و مقدار دهی شده است. حال برای تعریف نحوه نمایش داده‌ها در لیست باید مشخصه delegate را سفارشی و تعریف نماییم که تحت یک Component آن را ایجاد خواهیم کرد که به صورت زیر آورده شده است :

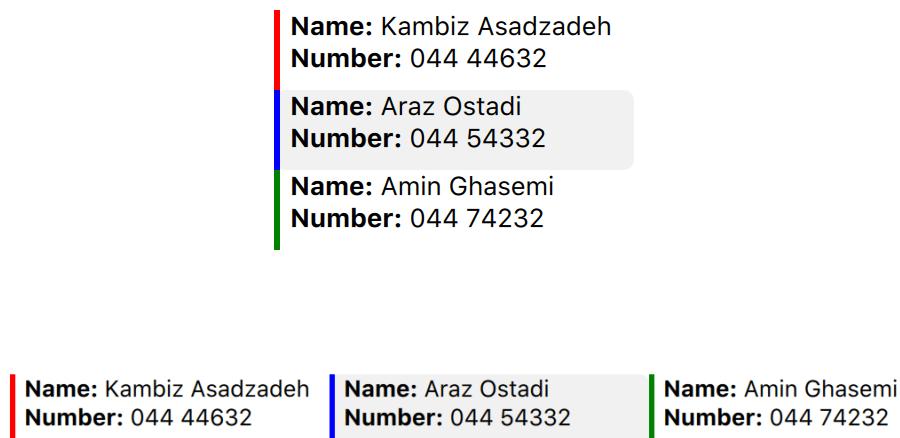
```
Component {  
    id: contactDelegate  
  
    Item {  
        width: 180; height: 40  
  
        Rectangle {  
            id: rec;  
            color : colorCode;  
            width: 3;  
            height: 40;  
        }  
  
        Column {  
            anchors.left: rec.right;  
            anchors.leftMargin: 5  
            Text { text: '<b>Name:</b> ' + name }  
            Text { text: '<b>Number:</b> ' + number }  
        }  
    }  
}
```

در این کامپوننت، ما از شیء مستطیل با مشخصه‌های رنگ و غیره به عنوان کد رنگ استفاده کرده‌ایم که مشخصه color آن برابر است با مشخصه .colorCode که آن نیز برابر است با مشخصه ContactModel در فایل که آن نیز برابر است با مشخصه .colorCode

در بخش بعد دو شیء از نوع متن برای نام و شماره در نظر گرفته شده است، هر یک از آن‌ها به ترتیب مقادیر name و number را دریافت و نمایش خواهند داد. در ادامه مقدار model لیست را برابر نام مدل ساخته شده قرار خواهیم داد:

```
ListView {  
    anchors.fill: parent  
    model: ContactModel {}  
    delegate: contactDelegate  
    highlight: Rectangle { color: "#f1f1f1"; radius: 5 }  
    focus: true  
    orientation: ListView.Horizontal  
}
```

نتیجهٔ نهایی کد نوشته شده به صورت زیر نمایان خواهد شد که در دو حالت افقی و عمودی با مشخصه orientation در ListView تنظیم شده است.



## نوع Loader

نوع Loader امکان بارگذاری پویا را برای زیر شاخه‌ها و یا آدرس یک کامپوننت را فراهم می‌سازد. به طور خلاصه وظیفهٔ بارگذاری پویای کامپوننت و یا فایل‌های QML را بر عهده دارد. بارگذاری Loader می‌تواند یک فایل را با استفاده از مشخصه source خود فراخوانی کند و یا می‌تواند با استفاده از مشخصه sourceComponent خود یک کامپوننت تعریف شده را بارگذاری کند. استفاده از آن برای از بین بردن تاخير در ایجاد یک کامپوننت و بارگذاری آن از طریق فایل بسیار مفید است.

نحوهٔ استفاده از آن به صورت زیر است :

```
Loader {  
    id: pageLoader;
```

```

anchors.centerIn: parent
}

MouseArea {
    anchors.fill: parent
    onClicked: pageLoader.source = "qrc:/Demo.qml"
}

```

در کد فوق با تعریف شناسه برای Loader آن را در رویداد کلیک ماوس با مشخصه source مسیر سند QML مورد نظر را داده ایم که حاوی کد زیر است:

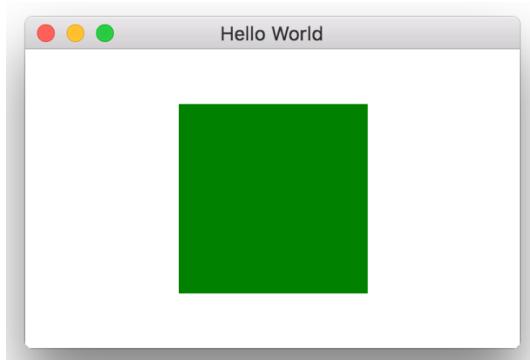
```

import QtQuick 2.9

Rectangle {
    width: 128
    height: 128
    color: "green"
}

```

در نهایت خروجی آن بعد از کلیک بر روی فرم به صورت زیر نمایان خواهد شد:



## نوع MouseArea

یک روش برای فعال سازی مدیریت رویدادهای ماوس است. رویداد لمسی تحت همین گزینه قابل استفاده است بنابراین محدود به رویدادهای ماوس نیست و در دستگاههای لمسی کاربرد ویژه ای دارد. نوع MouseArea یک آیتم نامرئی است که به طور معمول با دست زدن به آن در آیتمهای قابل مشاهده مورد استفاده قرار می گیرد. آیتمی که در محیط مربوط به آن قرار گیرد تحت تاثیر قرار خواهد گرفت. همچنین زمانی که مقدار enabled آن فعال باشد رخدادهای مربوط به آیتم با اولویت بالاتر فعال خواهند شد.

مشخصه `pressed` از نوع فقط خواندنی است و تنها هنگام فشرده شدن کلیک ماوس توسط کاربر در ناحیه مربوط به ایتم عمل می‌کند. مشخصه `containsMouse` نیز فقط خواندنی بوده و با حرکت اشاره گر ماوس بر روی آیتم مربوطه فعال می‌شود. همچنین باید توجه داشته باشید این خاصیت زمانی فعال است که خاصیت `diagram` با نام `hoverEnabled` فعال باشد. در غیر اینصورت در حالت کلیک شدن عمل خواهد کرد.

ساختار کلی رویداد ماوس به صورت زیر است :

```
MouseArea {
    onClicked: {
        //Do something...
    }
}
```

نوع	صفت
<code>Qt::MouseButtons</code>	<code>acceptedButtons</code>
<code>bool</code>	<code>containsMouse</code>
<code>bool</code>	<code>containsPress</code>
<code>Qt::CursorShape</code>	<code>courseShape</code>
<code>bool</code>	<code>enable</code>
<code>bool</code>	<code>hoverEnabled</code>
<code>bool</code>	<code>proposedAction</code>
<code>real</code>	<code>mouseX</code>
<code>real</code>	<code>mouseY</code>
<code>bool</code>	<code>pressed</code>
<code>MouseButtons</code>	<code>pressedButtons</code>
<code>bool</code>	<code>preventStealing</code>
<code>bool</code>	<code>propagateComposedEvents</code>
<code>bool</code>	<code>scrollGestureEnabled</code>
توضیح عملکرد سیگنال	سیگنال
زمانی که رویداد مرتبط با ماوس لغو شود این سیگنال ساطع خواهد گردید. البته باید به این نکته اشاره کرد که در طراحی‌های پیشرفته‌این سیگنال بسیار مفید است زیرا با ساطع شدن آن سریعاً سیگنال‌های دیگر در دسترس قرار می‌گیرند و سیستم می‌تواند نوع‌های <code>MouseArea</code> دیگر را کنترل نماید.	<code>canceled</code>
این سیگنال زمانی ساطع می‌شود که عمل کلیک شدن بر روی دکمه ماوس یا لمس بر روی ناحیه مربوطه صورت بگیر.	<code>clicked</code>
این سیگنال زمانی ساطع می‌شود که عمل دوبار کلیک شدن بر روی دکمه ماوس یا دو بار ضربه زدن به صورت لمسی بر روی ناحیه مربوطه صورت بگیر.	<code>doubleClicked</code>
این سیگنال زمانی ساطع می‌شود که اشاره گر ماوس در ناحیه مربوطه وارد شود. توجه داشته باشید که زمانی که مشخصه <code>hoverEnabled</code> فعال باشد این رویداد با وارد شدن اشاره گر در ناحیه کنترل می‌شود دی‌غیر این صورت با فشرده شدن ناحیه مربوطه سیگنال ساطع خواهد کرد.	<code>entered</code>
این سیگنال نیز دقیقاً بر عکس سیگنال <code>entered</code> عمل می‌کند.	<code>exited</code>
زمانی که مختصات و موقعیت اشاره گر ماوس تغییر کند این سیگنال ساطع خواهد شد.	<code>positionChanged</code>
این سیگنال زمانی ساطع می‌شود که به مدت طولانی بر روی ناحیه مربوطه عمل فشردن و نگه داشتن صورت بگیرد که معمولاً حدود ۸۰۰ میلی ثانیه است.	<code>pressAndHold</code>
زمانی که عمل فشرده شدن بر روی ناحیه صورت می‌گیرد این سیگنال ساطع می‌شود.	<code>pressed</code>
زمانی که عمل رها شدن بر روی یک ناحیه صورت بگیرد این سیگنال ساطع می‌شود.	<code>released</code>

در زیر مثالی آورده شده است که در آن تحت رویداد کلیک شدن رنگ شیء مستطیل تغییر خواهد کرد.

```
Rectangle {
    anchors.centerIn: parent
    width: 100; height: 100
    color: "green"

    MouseArea {
        anchors.fill: parent
        onClicked: { parent.color = 'red' }
    }
}
```

در مثال فوق شیء از نوع Rectangle با کلیک شدن توسط ماوس و یا لمس شدن بر روی صفحه لمسی بر روی ناحیه مورد نظر تغییر رنگ خواهد داد. تمامی رویدادهای موجود در این نوع در یک نوع ویژه با نام MouseEvent قرار دارند که می‌توان نسبت به حالت‌های مختلف آن اقدام به ایجاد رویداد کرد.

یکی از صفت‌های کاربردی این نوع acceptedButton نام دارد که وظیفه نگه داری و کنترل دکمه‌های راست را بر عهده دارد که تحت تمامی دکمه‌های مرتبط با ماوس در دسترس است و ساختار آن در QML به صورت زیر است :

```
MouseArea { acceptedButtons: Qt.LeftButton | Qt.RightButton }
MouseArea { acceptedButtons: Qt.AllButtons }
```

برای دسترسی به دکمه‌های موجود سمت چپ و راست بر روی ماوس از مقادیر Qt.RightButton و Qt.LeftButton استفاده می‌شود و در صورتی که نیاز است تمامی کلیدهای موجود بر روی ماوس مورد استفاده قرار بگیرند از مقدار Qt.AllButtons استفاده خواهد شد. البته باید توجه داشته باشید در صورت عدم مقدار دهی به مشخصه acceptedButtons مقدار آن همیشه برابر Qt.LeftButton خواهد بود.

زمانی که اشاره گر ماوس بر روی ناحیه مورد نظر وارد شود خاصیت containsMouse مورد استفاده قرار می‌گیرد. همچنین باید توجه داشته باشید که با غیرفعال بودن hoverEnabled این خاصیت فقط به فشرده شدن ناحیه محدود خواهد شد.

ساختار آن به صورت زیر است :

`mouseArea.containsMouse ? ارزش اول : ارزش دوم`

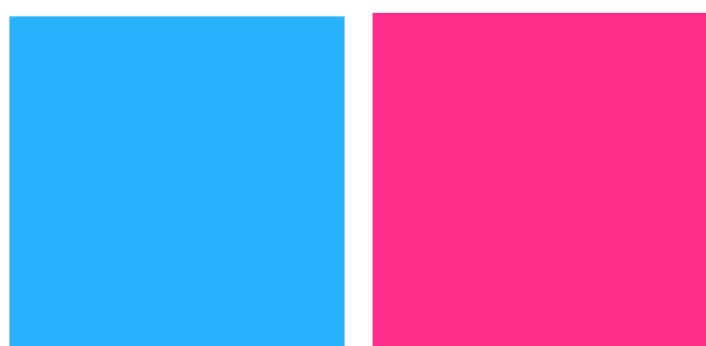
از این رویداد زمانی استفاده می‌شود که نیاز باشد با وارد شدن اشاره گر ماوس بر ناحیه مربوطه عملی صورت بگیرد برای مثال نیاز است با وارد شدن اشاره گر در ناحیه یک نوع Rectangle رنگ آن تغییر کند. برای اینکار ابتدا ساختار آن را به صورت زیر پیاده خواهیم کرد :

```
Rectangle {  
    width: 150; height: 150;  
    anchors.centerIn: parent  
    color: mouseArea.containsMouse ? "#FF338A" : "#33B3FF"  
  
    MouseArea {  
        id: mouseArea  
        anchors.fill: parent  
  
        hoverEnabled: true  
    }  
}
```

در مثال فوق دقت کنید که قابلیت hoverEnabled برابر است با true این مشخصه اگر غیرفعال باشد، خاصیت حرکت بر روی ناحیه مربوطه محدود به فشرده شدن ناحیه خواهد شد. برای استفاده از خاصیت ورود به ناحیه ماوس از مشخصه containsMouse استفاده می‌شود که با وارد کردن نام ناحیه به آن مشخصه دسترسی خواهید داشت.

نتیجه کد ذکر شده به صورت زیر موقع وارد شدن به ناحیه مربوطه تغییر خواهد کرد که در کد زیر به صورت یک دستور شرطی کوچک گنجانده شده است :

```
color: mouseArea.containsMouse ? "#FF338A" : "#33B3FF"
```



یک خاصیت دیگر با نام containsPressed نیز وجود دارد که مشابه containsMouse بوده و فقط با تفاوت فشرده شدن و وارد شدن بر روی محل مورد نظر عمل می‌کند. در واقع ترکیب وارد شدن و فشرده شدن بر روی ناحیه مورد نظر را توسط containsPressed می‌توان مدیریت کرد.

```
color: mouseArea.containsPressed ? "#FF338A" : "#33B3FF"
```

در رابطه با cursorShape باید گفت یکی از مهمترین و شاید جذابترین مشخصه‌هایی که در نوع MouseArea می‌توان یافت این مشخصه است که وظیفه مدیریت ظاهر (▷) اشاره گر ماوس را بر عهده دارد.

برای مثال ممکن است نیاز باشد اشاره گر پیشفرض را به یک اشاره گر سفارشی تبدیل کنید. برای مثال موقع وارد شدن بر یک ناحیه اشاره گر به صورت دیگری نمایش داده شود.

لیست اشکال مرتبط با اشاره گر به صورت زیر موجود است :

Qt.ArrowCursor	Qt.UpArrowCursor	Qt.CrossCursor	Qt.WaitCursor
Qt.IBeamCursor	Qt.SizeVerCursor	Qt.SizeHorCursor	Qt.SizeBDiagCursor
Qt.SizeFDiagCursor	Qt.SizeAllCursor	Qt.BlankCursor	Qt.SplitVCursor
Qt.SplitHCursor	Qt.PointingHandCursor	Qt.ForbiddenCursor	Qt.WhatsThisCursor
Qt.BusyCursor	Qt.OpenHandCursor	Qt.ClosedHandCursor	Qt.DragCopyCursor
Qt.DragMoveCursor	Qt.DragLinkCursor		

ساختار کلی و نحوه استفاده از آن به صورت زیر خواهد بود :

```
cursorShape: Qt.PointingHandCursor; acceptedButtons: Qt.NoButton
```

توجه داشته باشید که در این مشخصه مقدار acceptedButtons باید برابر None یا همان Qt.NoButton قرار بگیرد. در نهایت کد مرتبط با این ویژگی به صورت زیر است:

```
MouseArea {  
    id: mouseArea  
    anchors.fill: parent  
    hoverEnabled: true  
    cursorShape: Qt.PointingHandCursor;  
    acceptedButtons: Qt.NoButton  
}
```

مقدار پیشفرض cursorShape در صورت تعریف نشدن برابر با Qt.ArrowCursor است.

## MouseEvent نوع

نوع MouseEvent ارائه دهنده اطلاعات در رابطه با رویدادهای ماوس است. موقعیت ماوس می‌تواند از طریق مختصات x و y یافت شود. برای در دسترس بودن این رویداد از صفت button استفاده می‌شود.

نوع	صفت
bool	<b>accepted</b>
enumeration	<b>button</b>
int	<b>buttons</b>
int	<b>modifiers</b>
int	<b>source</b>
bool	<b>wasHeld</b>
int	<b>x</b>
int	<b>y</b>

تنظیم accepted بر روی true مانع ترویج رویدادهای ماوس در آیتم‌های زیرین خود می‌شود. همچنین در ویژگی button که دارای دکمه‌های راست - وسط - چپ است موجب عمل رویداد می‌شوند و شامل لیست زیر است:

- Qt.LeftButton
- Qt.RightButton
- Qt.MiddleButton

البته بهتر است در نظر داشته باشیم که مشخصه button نگه دارنده رویدادی است که توسط فشرده شده است که همانند دارای لیست زیر است:

- Qt.LeftButton
- Qt.RightButton
- Qt.MiddleButton

ویژگی modifiers پرچم (flag) مرتبط با صفحه کلید را که بلافاصله قبل از رویداد بر روی صفحه کلید ترکیب می‌شود را نگه می‌دارد. برای مثال ترکیب کلید Shift + کلیک راست بر روی ماوس این ویژگی شامل یک ترکیب بیتی به صورت زیر است:

- Qt.NoModifie
- Qt.ShiftModifier
- Qt.ControlModifier

- Qt.AltModifier
- Qt.MetaModifier
- Qt.KeypadModifier

در زیر مثالی آورده شده است که نشان می‌دهد با نگه داشتن کلید Shift بر روی صفحه کلید و فشردن کلیک راست بر روی ماوس عمل مورد نظر رخ خواهد داد. بنابراین ساختار کلی و نحوه استفاده از ویژگی modifiers به صورت زیر است:

```
MouseArea {
    id : mouseArea
    anchors.fill: parent
    onClicked: {
        if ((mouse.button === Qt.LeftButton) && (mouse.modifiers &
Qt.ShiftModifier))
            console.log("OKEY!")
    }
}
```

خاصیت source وظیفه نگه‌داری رویداد ماوس را بر عهده دارد، این ویژگی برای تشخیص دادن رویداد واقعی و مصنوعی ماوس مناسب است. زمانی که از یک دستگاه دیگری مانند دستگاه تبلت یا موبایل که دارای صفحه نمایش لمسی استفاده می‌شود این مشخصه به طور خود کار منبع رویداد را تشخیص می‌دهد و آن را توسط سیستم‌عامل و یا **Qt** هماهنگ می‌کند.

مقادیر این مشخصه یکی از موارد زیر می‌تواند باشد:

ثبت‌ها	توضیحات
Qt.MouseEventNotSynthesized	با ارزشترین مقدار را بر می‌گرداند. بر روی پلتفرم‌هایی که بر روی آن‌ها چنین اطلاعاتی در دسترس باشد مقدار واقعی یک رویداد ماوس را بر روی سیستم نشان می‌دهد.
Qt.MouseEventSynthesizedBySystem	نشان می‌دهد که رویداد ماوس از طرف یک صفحه لمسی یا تبلت توسط رویداد پلتفرم (سیستم‌عامل) ترکیب شده است.
Qt.MouseEventSynthesizedByQt	نشان می‌دهد که رویداد منتشر شده توسط صفحه لمسی و یک کنترل کننده مانند <b>Qt</b> ترکیب شده است.

Qt.MouseEventSynthesizedByApplication

نشان می‌دهد که رویداد ماوس توسط اپلیکیشن ترکیب شده است. این امکان را فراهم می‌سازد و اجازه می‌دهد تا رویدادهای تولید شده غیر واقعی توسط اپلیکیشن از طرف یک سیستم مانند Qt ترکیب شده‌اند.

مشخصه‌های x و y نیز مختصات محورهای مربوطه را نگه می‌دارند.

## MultiPointTouchArea نوع

نوع MultiPointTouchArea امکان لمس کردن چند نقطه را به صورت هم زمان فراهم می‌سازد. مشخصه Item::enabled برای فعال‌سازی و غیرفعال کردن قابلیت لمسی مورد استفاده قرار می‌گیرد. زمانی که غیرفعال باشد ناحیه مربوطه برای رویدادهای ماوس روشن سازی (قابل شناسایی) می‌شود.

به طور پیشفرض، ماوس روش لمسی یک نقطه را به کار می‌گیرد و آیتم‌های زیرین ناحیه لمسی رویدادهای ماوس را دریافت نمی‌کنند زیرا ناحیه لمسی آن‌ها را به کار می‌گیرد. اما در صورتی که خاصیت mouseEnabled برابر باشد با false رویدادهای مرتبط باهم ترکیب خواهند شد. مثال زیر یک کد نمونه برای ترکیب دو ناحیه به صورت چند لمسی است:

```
Rectangle {  
    width: 400; height: 400  
    MultiPointTouchArea {  
        anchors.fill: parent  
        touchPoints: [  
            TouchPoint { id: point1 },  
            TouchPoint { id: point2 }  
        ]  
    }  
    Rectangle {  
        width: 30; height: 30  
        color: "green"  
  
        x: point1.x  
        y: point1.y  
    }  
}
```

```

}

Rectangle {
    width: 30; height: 30
    color: "yellow"
    x: point2.x
    y: point2.y
}

```

با وارد کردن نوع MultiTouchArea نقطی که به صورت چند لمسی قابل استفاده است را به صورت پارامتر با شناسه‌های Rectangle به ترتیب point1 و point2 تعریف شده است که در آن دسترسی به نقاط محورهای x و y شیء‌های مربوطه فراهم شده است. این روش معمولاً در ساخت بازی و یا ترکیب چند لمسی اشیاء به کار می‌رود.

## NumberAnimation نوع

نوع NumberAnimation فراهم کننده تغییر و حرکت انیمیشن‌ها بر اساس مقادیر عددی است که یک نوع ارائه دهنده حرکات در جلوه‌های بصری بشمار می‌رود. ساختاری کلی آن به صورت زیر است:

```

NumberAnimation {
    from: 0;
    to: 1000;
}

```

نوع	صفت
real	from
real	to

صفات اصلی این نوع با نام‌های from و to است که به ترتیب مقادیر اولیه برای شروع انیمیشن و اتمام آن را بر عهده دارند. در زیر مثالی آورده شده است که در آن نوع NumberAnimation برای صفت محور x یک شیء مستطیل اختصاص داده شده است. این عمل حرکت در محور x را از مقدار اولیه آن به مقدار ۲۰۰ که مقدار نهایی خواهد بود آغاز می‌کند.

```

Rectangle {
    width: 100; height: 100
    radius: 100
    color: "green"
}

```

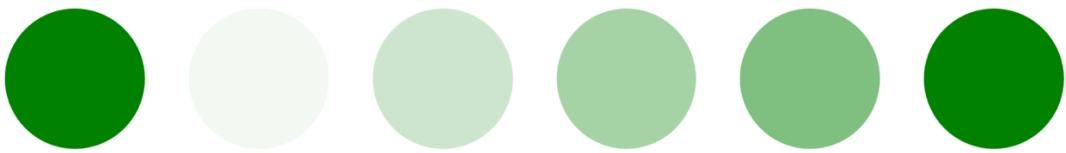
```

NumberAnimation on x {from:0; to: 200; }

}

```

در کد فوق شیء دایره در محور x با مقدار ۰ آغاز و مقدار ۲۰۰ پایانی به ترتیب در صفات from و to مشخص شده‌اند که از سمت چپ به راست حرکت خواهد کرد که نتیجه آن به صورت زیر است.



توجه داشته باشید که مانند هر انیمیشن دیگری نوع NumberAnimation می‌تواند با روش‌های مختلفی مورد استفاده قرار بگیرد که شامل، انتقال‌ها، رفتارها و مشخصه‌های منبع اعمال شود.

انیمیشن و انتقال در اسناد فناوری کیوت کوئیک روش‌های مختلفی را نشان می‌دهد. توجه داشته باشید که ممکن است در صورت تغییرات نامنظمه مقادیر شمارشی جلوه نمایشی به صورت صاف و یکنواخت تولید نشود. در این صورت می‌بایست به جای آن از SmoothedAnimation استفاده شود.

برخی از اعضای ویژه در این نوع که از PropertyAnimation ارث بری دارند به صورت زیر هستند که هر کدام جهت کار خاصی مورد استفاده قرار می‌گیرند.

توضیح (کاربرد)	نوع	صفت
مدت زمان اجرای انیمیشن را نگه می‌دارد (مقدار پیشفرض ۲۵۰ میلی ثانیه) است.	int	<b>duration</b>
یک نوع روان ساز (نوسان دار) جهت انیمیشن در حال اجرا تولید می‌کند.	real	<b>easy.amplitude</b>
یک نوع روان ساز (انحنا) جهت انیمیشن در حال اجرا تولید می‌کند.	list<real>	<b>easing.bezierCurve</b>
یک نوع روان ساز (پرتاپ) جهت انیمیشن در حال اجرا تولید می‌کند.	real	<b>easing.overshot</b>
یک نوع روان ساز (دوره‌ای) جهت انیمیشن در حال اجرا تولید می‌کند.	real	<b>easing.period</b>
جهت مشخص کردن شیوه روان سازی حرکت کاربرد دارد.	enumeration	<b>easing.type</b>
این صفت مشخص می‌کند که کدام آیتم تحت تاثیر انیمیشن قرار نگیرد.	list<Object>	<b>exclude</b>
این صفت مشخص می‌کند که کدام یک از (خاصیت‌ها) باید برای اجرا در انیمیشن در نظر گرفته شود.	string	<b>properties</b>
این صفت مشخص می‌کند که کدام (ویژگی - خصیصه) باید برای اجرا در انیمیشن در نظر گرفته شود.	string	<b>property</b>
این صفت مشخص می‌کند که کدام گزینه مورد نظر (شیء - شناسه از قبل تعریف شده) باید برای اجرا در انیمیشن در نظر گرفته شود.	Object	<b>target</b>

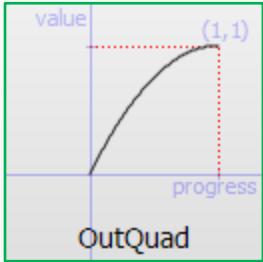
این صفت نیز همانند target مشخص می‌کند که کدام گزینه‌های مورد نظر (اشیاء - شناسه‌های از قبل تعریف شده) باید برای اجرا در انیمیشن در نظر گرفته شوند. با تفاوت اینکه به صورت دسته جمعی عمل خواهد کرد.	list<Object>	targets
--	--------------	---------

صفات زیر از نوع Animation ارث بری می‌کنند.

صفت	نوع	توضیح (کاربرد)
alwaysRunToEnd	bool	این صفت در صورتی که برابر با true باشد تا زمانی که انیمیشن به اتمام نرسیده باشد متوقف نخواهد شد.
loops	int	تعداد دفعات اجرایی انیمیشن را نگه می‌دارد.
paused	bool	اگر انیمیشن در حالت توقف قرار گرفته باشد برابر true خواهد بود.
running	bool	انیمیشن تعریف شده را به حالت اجرا در می‌آورد.
resume()	-	انیمیشن متوقف شده توسط paused را به حالت در می‌آورد.
complete()	-	انیمیشن در هر مرحله‌ای از اجرا باشد آن را به حالت پایان می‌رساند.
pause()	-	انیمیشن مربوطه را بدون اینکه به إتمام برسد متوقف می‌سازد.
restart()	-	انیمیشن مربوطه را مجدداً از اول اجرا می‌کند.
start()	-	انیمیشن تعریف شده را آغاز می‌کند.
started()	-	در صورتی که انیمیشن آغاز شده باشد از خود سیگنال ساطع می‌کند.
stop()	-	انیمیشن را به طور کامل متوقف می‌کند.
stoped()	-	در صورتی که انیمیشن به طور کامل متوقف شده باشد از خود سیگنال ساطع خواهد کرد.

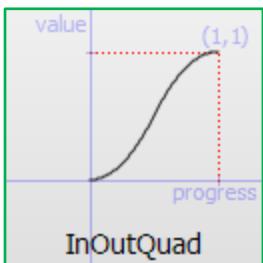
در ادامه جدول انواع شیوه‌های روان سازی انیمیشن آمده است که به صورت زیر است:

<p>Linear</p>	<p>روان سازی به صورت منحنی به صورت یک تابع خطی (با سرعت ثابت).</p> <p>Easing.Linear</p>
<p>InQuad</p>	<p>روان سازی به صورت منحنی برای یک تابع درجه دوم (شتاب از صفر).</p> <p>Easing.InQuad</p>



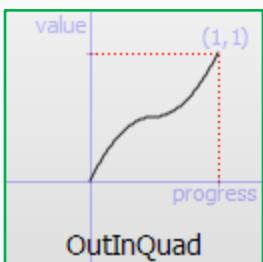
روان سازی به صورت منحنی برای یک تابع درجه دوم (کاهش  
شتاب به طرف صفر)

**Easing.OutQuad**



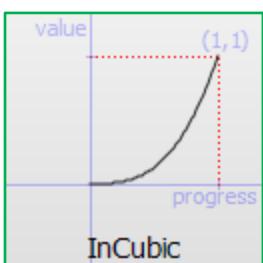
روان سازی به صورت منحنی برای یک تابع درجه دوم (افزایش  
سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutQuad**



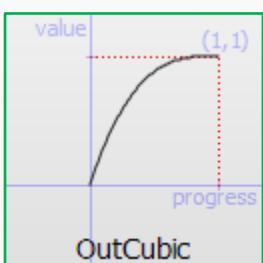
روان سازی به صورت منحنی برای یک تابع درجه دوم (کاهش  
سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه  
انتهایها)

**Easing.OutInQuad**



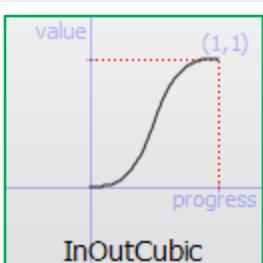
روان سازی به صورت منحنی برای یک تابع درجه سوم (کاهش  
شتاب از سرعت صفر)

**Easing.InCubic**



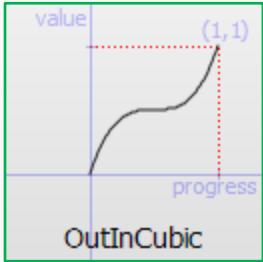
روان سازی به صورت منحنی برای یک تابع درجه سوم (کاهش  
شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutCubic**



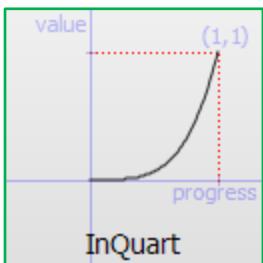
روان سازی به صورت منحنی برای یک تابع درجه سوم (افزایش  
سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutCubic**



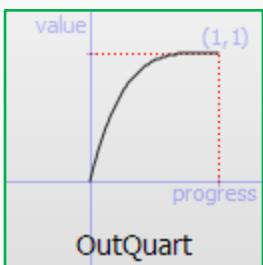
روان سازی به صورت منحنی برای یک تابع درجه سوم (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتها)

**Easing.OutInCubic**



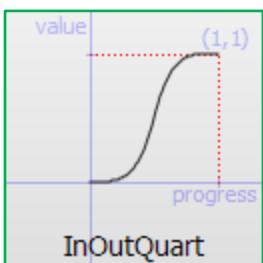
روان سازی به صورت منحنی برای یک تابع درجه چهارم (شتاب از صفر)

**Easing.InQuart**



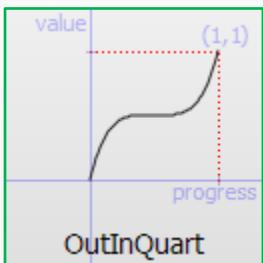
روان سازی به صورت منحنی برای یک تابع درجه چهارم (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutQuart**



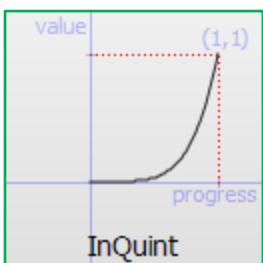
روان سازی به صورت منحنی برای یک تابع درجه چهارم (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتها)

**Easing.InOutQuart**



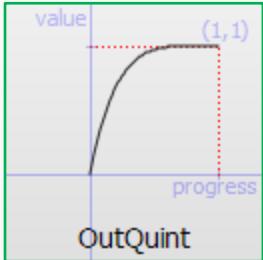
روان سازی به صورت منحنی برای یک تابع درجه چهارم (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتها)

**Easing.OutInQuart**



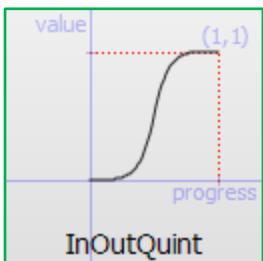
روان سازی به صورت منحنی برای یک تابع درجه پنجم (شتاب از صفر)

**Easing.InQuint**



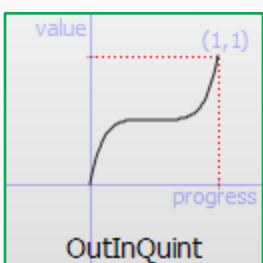
روان سازی به صورت منحنی برای یک تابع درجه پنجم (کاهش  
شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutQuint**



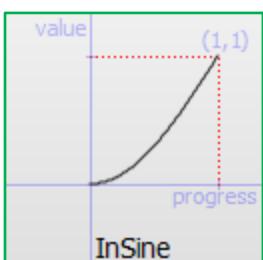
روان سازی به صورت منحنی برای یک تابع درجه پنجم (افزایش  
سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutQuint**



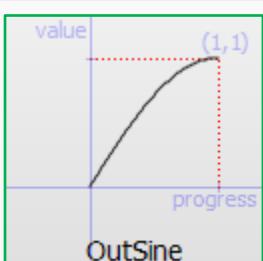
روان سازی به صورت منحنی برای یک تابع درجه پنجم (کاهش  
سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه  
انتهایها)

**Easing.OutInQuint**



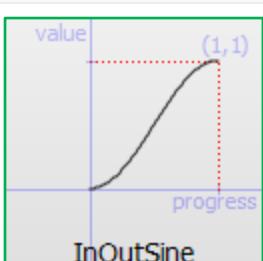
روان سازی به صورت منحنی برای یک تابع سینوسی (شتاب از  
صفر)

**Easing.InSine**



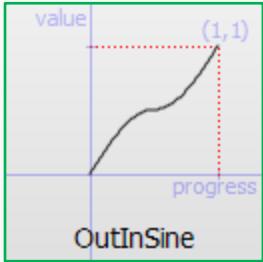
روان سازی به صورت منحنی برای یک تابع سینوسی (کاهش  
شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutSine**



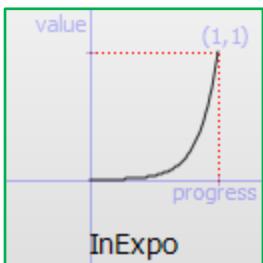
روان سازی به صورت منحنی برای یک تابع سینوسی (افزایش  
سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutSine**



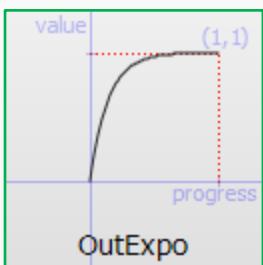
روان سازی به صورت منحنی برای یک تابع سینوسی (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتها)

**Easing.OutInSine**



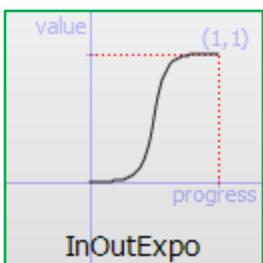
روان سازی به صورت منحنی برای یک تابع نمایی (شتاب از صفر)

**Easing.InExpo**



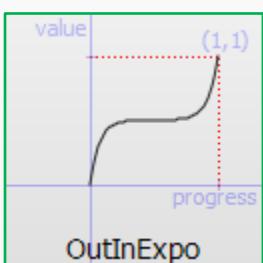
روان سازی به صورت منحنی برای یک تابع نمایی (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutExpo**



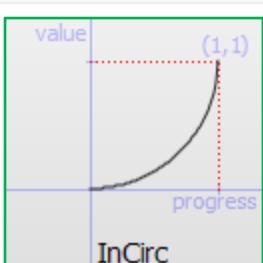
روان سازی به صورت منحنی برای یک تابع نمایی (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتها)

**Easing.InOutExpo**



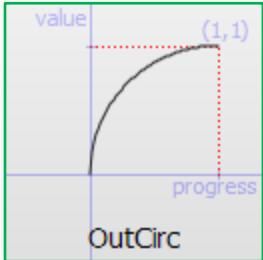
روان سازی به صورت منحنی برای یک تابع نمایی (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتها)

**Easing.OutInExpo**



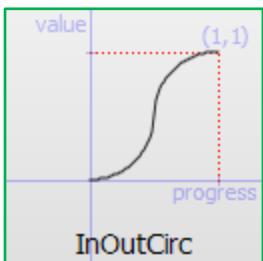
روان سازی به صورت منحنی برای یک تابع دایره‌ای (شتاب از صفر)

**Easing.InCirc**



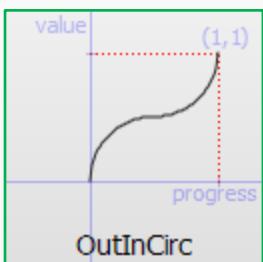
روان سازی به صورت منحنی برای یک تابع دایره‌ای (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutCirc**



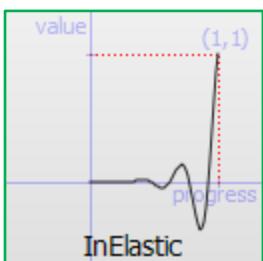
روان سازی به صورت منحنی برای یک تابع دایره‌ای (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutCirc**



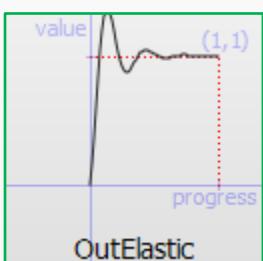
روان سازی به صورت منحنی برای یک تابع دایره‌ای (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتهایها)

**Easing.OutInCirc**



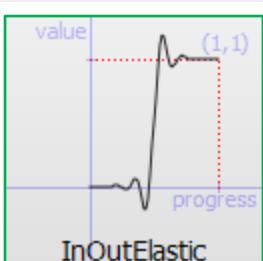
روان سازی به صورت منحنی برای یک تابع الاستیک (میرایی نمایی تا زمان موج‌ها) شتاب از صفر راس دامنه می‌تواند با *amplitudeparameter* تعیین شود و دامنه نوسان می‌تواند با *periodparameter* تعیین شود.

**Easing.InElastic**



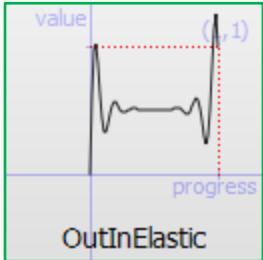
روان سازی به صورت منحنی برای یک تابع الاستیک (میرایی نمایی تا زمان موج‌ها) (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutElastic**



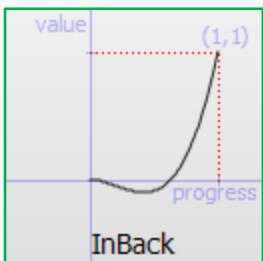
روان سازی به صورت منحنی برای یک تابع الاستیک (میرایی نمایی تا زمان موج‌ها) (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهایها)

**Easing.InOutElastic**



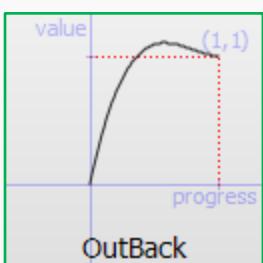
روان سازی به صورت منحنی برای یک تابع الاستیک ( میرایی نمایی تا زمان موج ها ) (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتهای)

**Easing.OutInElastic**



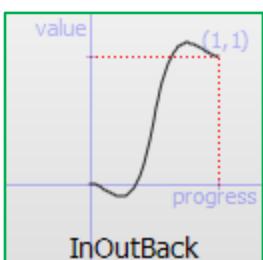
روان سازی به صورت منحنی برای یک تابع جهشی  $(s+1)*t^3 - s*t^2$  ( میرایی نمایی تا زمان موج ها ) شتاب از صفر

**Easing.InBack**



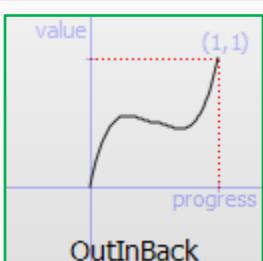
روان سازی به صورت منحنی برای یک تابع جهشی  $(s+1)*t^3 - s*t^2$  ( میرایی نمایی تا زمان موج ها ) (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر )

**Easing.OutBack**



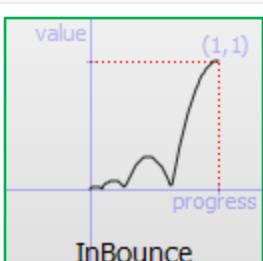
روان سازی به صورت منحنی برای یک تابع جهشی  $(s+1)*t^3 - s*t^2$  ( میرایی نمایی تا زمان موج ها ) (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهای)

**Easing.InOutBack**



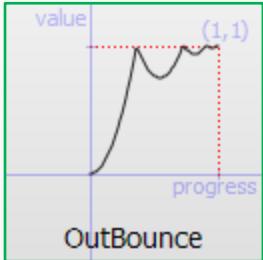
روان سازی به صورت منحنی برای یک تابع جهشی  $(s+1)*t^3 - s*t^2$  ( میرایی نمایی تا زمان موج ها ) (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتهای)

**Easing.OutInBack**



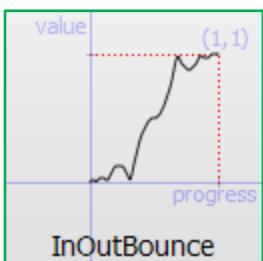
روان سازی به صورت منحنی برای یک تابع پرشی پارabolیک ( میرایی نمایی تا زمان موج ها ) شتاب از صفر

**Easing.InBounce**



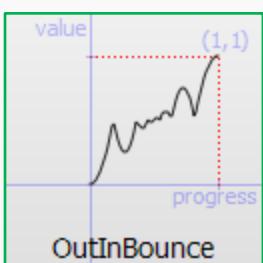
روان سازی به صورت منحنی برای یک تابع پرشی پارabolیک (میرایی نمایی تا زمان موج ها) (کاهش شتاب از بالاترین سرعت به طرف سرعت صفر)

**Easing.OutBounce**



روان سازی به صورت منحنی برای یک تابع پرشی پارabolیک (میرایی نمایی تا زمان موج ها) (افزایش سرعت تا نیمه راه و سپس کاهش سرعت تا رسیدن به نقطه انتهای)

**Easing.InOutBounce**



روان سازی به صورت منحنی برای یک تابع پرشی پارabolیک (میرایی نمایی تا زمان موج ها) (کاهش سرعت تا نیمه راه و سپس افزایش سرعت تا رسیدن به نقطه انتهای)

**Easing.OutInBounce**

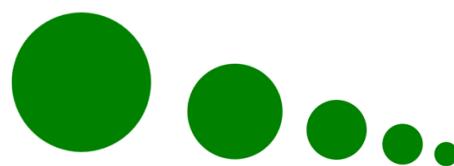
سایر توابع سفارشی با دستور easing.bezierCurve تولید می‌شوند.

**Easing.Bezier**

برای مثال با استفاده از خاصیت Easing.OutBounce شیء مورد نظر را نمایان خواهیم ساخت که در خاصیت ScaleAnimator برای مقایس پذیری صورت گرفته است. در این مثال شیء با به کار گیری خاصیت OutBounce مقدار مقایس پذیری آن از ۰ به مقدار ۱ تغییر یافته و نمایان خواهد یافت.

```
Rectangle {
    id: testRec
    width: 100; height: 100
    radius: 100
    color: "green"

    ScaleAnimator {
        target: testRec;
        from: 0;
        to: 1;
        duration: 1000
        running: true
        easing.type: Easing.OutBounce;
    }
}
```



## نوع OpacityAnimator

نوع OpacityAnimator برای شفافیت آیتم مورد استفاده قرار می‌گیرد. انواع انیمیشن‌های تعریف شده QML با انواع استاندارد متفاوت هستند. ارزش (مقدار) Item::opacity بعد از به اتمام رسیدن انیمیشن به روز رسانی می‌شود. ساختار آن به صورت زیر است که شامل خاصیت‌های target برای شیء مورد نظر جهت اعمال انیمیشن و خاصیت‌های from برای مقدار اولیه برای شروع و to برای مقدار نهایی برای اتمام با دو خاصیت duration و running که به ترتیب جهت مشخص کردن مقدار زمان اجرا و وضعیت اجرا شدن آن در نظر گرفته شده است.

```
OpacityAnimator {
    target: object;
    from: 0;
    to: 1;
    duration: 2000
    running: true
}
```

بر فرض اینکه لازم است یک شیء مستطیل را توسط نوع `OpacityAnimator` شفاف سازی نماییم. بنابراین کد آن به صورت زیر خواهد بود:

```
Rectangle {  
    id: opacityBox  
    width: 150  
    height: 150  
    color: "#009688"  
  
    OpacityAnimator {  
        target: opacityBox;  
        from: 0;  
        to: 1;  
        duration: 2000  
        running: true  
    }  
}
```

در کد فوق شکل مورد نظر در ابعاد ۱۵۰ در ۱۵۰ پیکسل تولید و توسط انیمیتور `OpacityAnimator` موقع اجرا شدن در بازه زمانی ۲ ثانیه از مقدار ۰ به مقدار ۱ تغییر شفافیت خواهد داد. توجه داشته باشید که مشخصه `target` برابر شناسه شیء مورد نظر برای اعمال انیمیشن است.



البته روش دومی هم برای اعمال انیمیشن برای اشیاء موجود است که بدون هدف گذاری توسط مشخصه `target` صورت می‌گیرد که شکل کلی آن به صورت زیر است:

### `OpacityAnimator on opacity { ... }`

```
Rectangle {  
    id: opacityBox  
    width: 150  
    height: 150  
    color: "#009688"  
  
    OpacityAnimator on opacity{  
        from: 0;  
        to: 1;  
        duration: 2000  
    }  
}
```

```
    }  
}
```

در این روش انیمیتور OpacityAnimator مستقیماً اعمال وضعیت را بر روی خاصیت opacity مرتبط با شیء مربوطه اعمال می‌کند. این روش زمانی توصیه می‌شود که قرار است هریک از اشیاء دارای انیمیشن منحصر به فرد باشند و نیازی به استفاده مشخصه target نباشد.

## نوع ParallelAnimation

نوع ParallelAnimation موجب می‌شود تا انیمیشن‌ها در QML به صورت موازی اجرا شوند. انواع SequentialAnimation و ParallelAnimation اجازه می‌دهند تا انیمیشن‌های متعددی در کنار هم اجرا شوند. انیمیشن‌های تعریف شده در یک SequentialAnimation یکی پس از دیگری اجرا می‌شوند، در حالی که انیمیشن‌های تعریف شده در یک ParallelAnimation در همان زمان اجرا خواهند شد.

ساختار آن به صورت زیر است:

```
ParallelAnimation {  
    running: true  
    ...  
}
```

در زیر مثالی آورده شده است که به روش موازی شیء مورد نظر در قالب یک انیمیشن تغییر موقعیت خواهد داد.

```
Rectangle {  
    id: rect  
    width: 150  
    height: 150  
    color: "#009688"  
  
    ParallelAnimation {  
        running: true  
        NumberAnimation {  
            target: rect;  
            property: "x";  
            to: 50;  
            duration: 1000  
        }  
        NumberAnimation {  
            target: rect;  
            property: "y";  
            to: 50;  
            duration: 1000  
        }  
    }  
}
```

```

    target: rect;
    property: "y";
    to: 50;
    duration: 1000
}
}

}

```

کد فوق شامل یک شیء Rectangle با شناسه rec است. در نظر داریم آن را در محور x و y به حرکت در آوریم که در بازه زمانی ۱ ثانیه اتفاق خواهد افتاد.

همانطور که در صفحه قبل اشاره شده است NumberAnimation برای به حرکت در آوردن شیء مورد استفاده قرار می‌گیرد که در این بخش با مقدار دهی مشخصه property برای مشخص کردن اینکه کدام مشخصه از محورها می‌بایست حرکت یابد به کار رفته است. هدف آن (target) برابر شناسه شیء و در نهایت موقعیت آن با مقدار ۵۰ در محور x و y تغییر می‌یابد.



نوع	صفت
int	duration
real	easing.amplitude
list<real>	easing.bezierCurve
real	easing.overshoot
real	easing.period
enumeration	easing.type

list<Object>	<b>exclude</b>
variant	<b>from</b>
string	<b>properties</b>
string	<b>property</b>
Object	<b>target</b>
list<Object>	<b>targets</b>
variant	<b>to</b>

## نوع GraphicInfo

نوع ارتباط اطلاعات مرتبط با بک‌اند نمودارهای صحنه‌ای را فراهم می‌کند. از مشخصه‌های آن می‌توان به minorVersion، renderableType و profile و majorVersion اشاره کرد. البته با توجه به روز رسانی‌های مرتبط با این نسخه از کتاب سعی شده است با آخرین تغییرات ممکن هماهنگ شود. توجه داشته باشید که به جای OpenGLInfo از نسخه ۵.۸ و ۵.۹ به بعد از **GraphicsInfo** استفاده خواهیم کرد. بنابراین مثال‌های فوق را نیز بر اساس نوع جدید GraphicsInfo ارائه می‌دهیم.

نوع	صفت
enumeration	<b>api</b>
int	<b>majorVersion</b>
int	<b>minorVersion</b>
enumeration	<b>profile</b>
enumeration	<b>renderableType</b>
enumeration	<b>shaderCompilationType</b>
enumeration	<b>shaderSourceType</b>
enumeration	<b>shaderType</b>

برای مثال می‌خواهیم بدانیم برنامه ما در حال حاضر بر پایه کدام رابط برنامه‌نویسی (API) گرافیکی کار می‌کند. بنابراین با توجه به ویژگی‌های موجود در صفت api به آن دسترسی خواهیم داشت که هم در بخش **فرانت‌اند** و هم بک‌اند در دسترس هستند و مقادیر ممکن آن عبارتند از:

- نوع **GraphicsInfo.Unknown** درصورتی که مقدار فعالی از نمودارهای صحنه‌ای مشخص نباشد بازگردنده خواهد شد.

- نوع **GraphicsInfo.Software** درصورتی این مقدار فعال باشد مشخص می‌کند که از موتور تولید تصاویر نرم‌افزاری بر پایه **QPainter** استفاده می‌کند.

- نوع **GraphicsInfo.OpenGL** درصورتی که از نوع OpenGL و یا ES استفاده شود بازگردنده خواهد شد.

- نوع **GraphicsInfo.Direct3D12** درصورتی که از نوع Direct3D12 استفاده شود بازگردنده خواهد شد (پلتفرم ویندوز).

- نوع **GraphicsInfo.OpenVG** درصورتی که از نوع گرافیک برداری دو بعدی استفاده شود بازگردنده خواهد شد نسخه (در دسترس از نسخه ۵.۹ کیوت).

بنابراین می‌توانیم تحت یک دستور شرطی از اطلاعات مرتبط با آن استفاده نماییم که در ادامه به دو نمونه کد QML و C++ اشاره کرده‌ایم.

```
switch (QSGRendererInterface::GraphicsApi()) {
case QSGRendererInterface::Unknown:
    qDebug() << "An unknown graphics API is in use";
```

```

        break;
    case QSGRendererInterface::Software:
        qDebug() << "The Qt Quick 2D Renderer is in use";
        break;
    case QSGRendererInterface::OpenGL:
        qDebug() << "OpenGL ES 2.0 or higher";
        break;
    case QSGRendererInterface::Direct3D12:
        qDebug() << "Direct3D 12";
        break;
    case QSGRendererInterface::OpenVG:
        qDebug() << "OpenVG";
        break;
    default:
        break;
}

```

در کد فوق از روش شرطی برای شمارش انواع شمارشی از نوع `Enum` استفاده شده است که در صورت برابری مقادیر آن پیغام مرتبط با هریک از آنها چاپ خواهد شد. همچنین جهت دریافت اطلاعات ویرایشی (نسخه) سیستم گرافیکی طبق دستور زیر می‌توان از مشخصه‌های `minorVersion` و `majorVersion` استفاده کرد:

```

Text {
    id:version
    text: GraphicsInfo.majorVersion + "." + GraphicsInfo.minorVersion;
}

```

نوع مقادیر مرتبط با نوع پیکربندی را باز می‌گرداند که در نمونه زیر به آن اشاره شده است:

```

var profileString = ""

switch (GraphicsInfo.profile) {
    case GraphicsInfo.OpenGLNoProfile:
        profileString = "OpenGL version is lower than 3.2 or OpenGL is not in use."
        break
    case GraphicsInfo.OpenGLCoreProfile:
        profileString = "Functionality deprecated in OpenGL version 3.0 is not
available."
        break
    case GraphicsInfo.OpenGLCompatibilityProfile:
        profileString = "Functionality from earlier OpenGL versions is available."
        break
}

```

نوع `renderableType` روش تولید تصویر را برمی‌گرداند که تنها در نوع `OpenGL` قابل استفاده است. `GraphicsInfo.SurfaceFormatUnspecified` به طور پیشفرض بوده و زمانی که متده تولید یافت نشود مقدار آن برگشت داده خواهد شد.

در زمان مشخص بودن نوع تولید تصویر بر اساس `OpenGL` و دیگر رابطه‌های گرافیکی مقدار خود را برمی‌گرداند. `GraphicsInfo.SurfaceFormatOpenGLES` زمانی مقدار بازگشتی خواهد بود که نوع `OpenGL ES` مشخص شده باشد.

```

var renderableTypeString = "";
switch (GraphicsInfo.renderableType) {

    case GraphicsInfo.SurfaceFormatUnspecified:
        renderableTypeString = "Unspecified rendering method";
        break
    case GraphicsInfo.SurfaceFormatOpenGL:
        renderableTypeString = "Desktop OpenGL or other graphics API";
        break
    case GraphicsInfo.SurfaceFormatOpenGLES:
        renderableTypeString = " OpenGL ES";
        break
}

```

ویژگی **shaderCompilationType** شامل یک ماسک از تدوین سایه زنی است که در برنامه‌های Qt Quick مورد استفاده قرار می‌گرد. که اگر از OpenGL استفاده شود ارزش یا مقدار آن برابر با **GraphicsInfo.RuntimeCompilation** به روش سنتی جهت استفاده از سایه زنی مشخص خواهد شد. همچنین در صورتی که بک‌اند آن به جز نوع OpenGL مشخص شود نوع Offline درنظر گرفته خواهد شد.

```

var shader = "";
switch (GraphicsInfo.shaderCompilationType) {

    case GraphicsInfo.RuntimeCompilation:
        renderableTypeString = "Runtime";
        break
    case GraphicsInfo.OfflineCompilation:
        renderableTypeString = "Offline";
        break
}

```

در صورت استفاده در بک‌اند نمونه کد C++ به طور زیر خواهد بود:

```

switch (QSGRendererInterface::ShaderCompilationType()) {
    case QSGRendererInterface::RuntimeCompilation:

        qDebug() << "Shader source can be provided as a string in the
corresponding properties of ShaderEffect";
        break;
    case QSGRendererInterface::OfflineCompilation:
        qDebug() << "Local or resource files containing shader source code are
supported";
        break;
    default:
        qDebug() << "ShaderCompilationType not found!";
        break;
}

```

ویژگی **shaderSourceType** شامل یک ماسک از تدوین منابعی که پشتیبانی می‌شوند است.

```

var sourceType = "";
switch (GraphicsInfo.shaderSourceType) {

    case GraphicsInfo.ShaderSourceString:
        sourceType = "ShaderSourceString";
        break
    case GraphicsInfo.ShaderSourceFile:
        sourceType = "ShaderSourceFile";
        break
    case GraphicsInfo.ShaderByteCode:
        sourceType = "ShaderByteCode";
        break
}

```

در صورت استفاده در بَک‌اِند نمونه کد C++ به طور زیر خواهد بود:

```

switch (QSGRendererInterface::ShaderSourceTypes()) {
    case QSGRendererInterface::ShaderSourceString:
        qDebug() << "Shader source can be provided as a string in the
corresponding properties of ShaderEffect";
        break;
    case QSGRendererInterface::ShaderSourceFile:
        qDebug() << "Local or resource files containing shader source code are
supported";
        break;
    case QSGRendererInterface::ShaderByteCode:
        qDebug() << "Local or resource files containing shader bytecode are
supported";

        break;
    default:
        qDebug() << "Source not found!";
        break;
}

```

همچنین ویژگی shaderType نوع سایه زنی را بازمی‌گرداند که شامل .UnknownShadingLanguage اس. HLSL است.

```

var shadertype = "";
switch (GraphicsInfo.UnknownShadingLanguage) {

    case GraphicsInfo.GLSL:
        renderableTypeString = "GLSL or GLSL ES";
        break
    case GraphicsInfo.HLSL:
        renderableTypeString = "HLSL";
        break
}

```

در صورت استفاده در بَک‌اِند نمونه کد C++ به طور زیر خواهد بود:

```

switch (QSGRendererInterface::ShaderType) {
    case QSGRendererInterface::UnknownShadingLanguage:

```

```

qDebug() << "Not yet known due to no window and scenegraph associated";
break;
case QSGRendererInterface::GLSL:
qDebug() << "GLSL or GLSL ES";
break;
case QSGRendererInterface::HLSL:
qDebug() << "HLSL";
break;
default:
qDebug() << "Shader type not found!";
break;
}

```

## نوع ParentAnimation

از نوع ParentAnimation زمانی استفاده می‌شود که بخواهیم یک آیتم را با تغییر والد آن به حرکت در آوریم. در واقع در صورت حرکت یک شیء آن به عنوان بخشی از والد ترکیب خواهد شد.

به مثال زیر توجه کنید:

```

Rectangle {
    id: redRect
    width: 100; height: 100
    color: "red"
}

Rectangle {
    id: blueRect
    x: redRect.width
    width: 50; height: 50
    color: "blue"
    states: State {
        name: "reparented"
        ParentChange { target: blueRect; parent: redRect; x: 10; y: 10 }
    }
    transitions: Transition {
        ParentAnimation {
            NumberAnimation { properties: "x,y"; duration: 1000 }
        }
    }
    MouseArea { anchors.fill: parent; onClicked: blueRect.state =
"reparented" }
}

```

زمانی که حرکت در داخل ParentAnimation صورت می‌گیرد مشخصه‌های x و y آن با اطمینان خاطر اینکه در داخل والد تغییر خواهد یافت تضمین می‌شود. در واقع تضمینی برای آن است که حرکت در داخل شیء والد صورت خواهد گرفت.

## نوع ParentChange

نوع ParentChange یک آیتم را در زمان حرکت از لحاظ ظاهری، موقعیت، چرخش و مقیاس بر روی صفحه حفظ نموده و آن را در والد خود نمایش می‌دهد.

```

Rectangle {
    id: redRect
    width: 100; height: 100
    color: "red"
}

Rectangle {
    id: greenRect
    x: redRect.width
    width: 50; height: 50
    color: "green"

    states: State {
        name: "reparented"
        ParentChange { target: greenRect; parent: redRect; x: 10; y: 10 }
    }

    MouseArea { anchors.fill: parent; onClicked: greenRect.state =
"reparented" }
}

```

به طور کلی این ویژگی مشخص می‌کند که چطور باید حالت reparented اجرا شود. در این مثال مختصات x و y مقدار دهی شده و زمان کلیک بر روی greenRect وضعيت reparented مشخص شده است اجرا می‌شود.

## نوع Path

یک نوع Path خود از یک یا چند نوع مسیر دهی تشکیل شده است که شامل PathLine, PathQuad, PathCubic, PathArc, PathCurve و PathSvg است. همچنین توجه داشته باشید که فاصله بین آیتم‌ها در نوع Path می‌تواند توسط PathPercent تنظیم شود. نوع Path می‌تواند شامل اشیاء مسیر زیر نیز باشد:

- یک خط مستقیم به موقعیت اعمال می‌کند.
  - یک منحنی نوع درجه دوم به یک نقطه کنترل اعمال می‌کند.
  - یک مکعب منحنی با دونقطه‌ی کنترل به موقعیت اعمال می‌کند.
  - یک قوس همراه با یک شعاع به موقعیت اعمال می‌کند.
  - یک مسیر از نوع داده رشتہ‌ای در قالب SVG مشخص می‌کند.
  - یک نقطه در منحنی Catmull-Rom اعمال می‌کند.
  - یک ویژگی در یک مسیری از موقعیت اعمال می‌کند.
  - روشی برای گسترش بخش‌های خارجی آیتم‌های مسیر مشخص می‌کند.
- برای درک بهتر به مثال زیر توجه کنید:

```

Canvas {
    width: 500; height: 500
    contextType: "2d"
}

```

```

Path {
    id:myPath
    startX: 50; startY: 50
    PathLine { x: 100; y: 200 }
}

onPaint: {
    context.strokeStyle = Qt.rgba(.5,.3,.3);
    context.path = myPath;
    context.stroke();
}
}

```

در مثال فوق توسط نوع Path و روش رسم بر روی آن به صورت PathLine مشخص شده است که در محور x به مقدار ۱۰۰ و y به مقدار ۲۰۰ مشخص شده است. مشخصه startX نقطه سر آغاز برای رسم خط را مشخص می‌کند. بنابراین مسیری که با این دستور مشخص شده است توسط متده است. در آیتم onPaint رسم خواهد شد که به صورت زیر خواهد بود.



نتیجه فوق به دلیل استفاده از نوع PathLine یک خط راست و مستقیم را رسم می‌کند. بنابراین اگر تعداد زیادی از آن‌ها را باهم ترکیب در مختصات مشخصی ترکیب کنیم به صورت زیر خواهیم داشت:

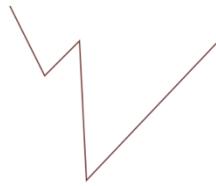
```

Canvas {
    width: 500; height: 500
    contextType: "2d"

    Path {
        id:myPath
        startX: 50; startY: 50
        PathLine { x: 75; y: 100 }
        PathLine { x: 100; y: 75 }
        PathLine { x: 105; y: 175 }
        PathLine { x: 200; y: 75 }
    }

    onPaint: {
        context.strokeStyle = Qt.rgba(.5,.3,.3);
        context.path = myPath;
        context.stroke();
    }
}

```



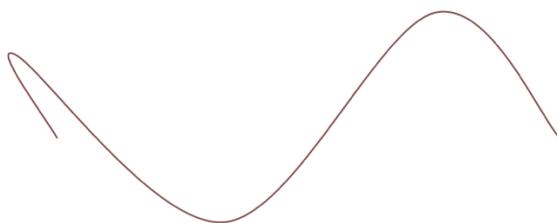
اگر دقت کنید تمامی مسیرهای طی شده توسط PathLine و چاپ بر روی آنها صورت گرفته اند اما با این حال دارای منحنی خاصی نیستند. در صورتی که نیاز باشد به طور منحنی خطی را رسم کنیم بهتر است از نوع PathCurve استفاده کنیم که مثالی در این رابطه به صورت زیر خواهیم داشت:

```
Canvas {
    width: 500; height: 500
    contextType: "2d"

    Path {
        id:myPath
        startX: 100; startY: 100

        PathCurve { x: 75; y: 50 }
        PathCurve { x: 200; y: 150 }
        PathCurve { x: 325; y: 25 }
        PathCurve { x: 400; y: 100 }
    }

    onPaint: {
        context.strokeStyle = Qt.rgba(.5,.3,.3);
        context.path = myPath;
        context.stroke();
    }
}
```



با استفاده از نوع PathCurve می‌توانیم با اختصاص دادن مقادیر مرتبط با محورهای X و Y برای هربار رسم یک خط به طور منحنی رسمی نماییم که نقطه شروع محورهای X و Y با startX و startY مشخص می‌شود.

نکته: برای تعریف مختصات x و y برای نقاط پایانی نسبت به نقاط شروع کافی است از ویژگی relativeY و relativeX استفاده کنید. در غیر اینصورت استفاده از x و y به طور عادی تنها نقاطی را برای پایان خط تعریف می‌کند.

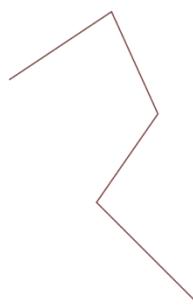
مثال:

```
Path {
    id:myPath
    startX: 50; startY: 50
    PathLine { relativeX: 75; relativeY: -50 }
```

```

    PathLine { relativeX: 34; relativeY: 75 }
    PathLine { relativeX: -45; relativeY: 65 }
    PathLine { relativeX: 75; relativeY: 75 }
}

```



مختصات تعریف شده به صورت نسبی تصویر فوق را تولید می‌کند.

در نوع PathSVG جهت ترسیم اشکال به روش برداری صورت می‌گیرد که به طور زیر یک مثال از آن آورده شده است:

```

Canvas {
    width: 500; height: 500
    contextType: "2d"

    Path {
        id:myPath
        startX: 20; startY: 100
        PathSvg { path: "L 150 50 L 100 150 z" }
    }

    onPaint: {
        context.strokeStyle = Qt.rgba(.5,.3,.3);
        context.path = myPath;
        context.stroke();
    }
}

```

باتوجه به کد ذکر شده مشخص است که مشخصه مسیر آن شامل دستور L و z است. به طور خلاصه دستور L در نوع SVG به معنی این است که خطی را از نقطه شروع تا پایان رسم کند.

برای مثال نقطه شروع در این کد از ۱۵۰ و نقطه پایان آن ۵۰ است. در ادامه نقطه شروع مجدد ضلیع دوم ۱۰۰ و پایان آن ۱۵۰ است. دستور z نیز به معنی خاتمه دهنده رسم عمل می‌کند.

دستور	پارامتر	نام	توضیحات
z یا Z	ندارد	closepath	بستان زیر خط فعلی با رسم یک خط راست از نقطه فعلی به نقطه فعلی دیگر که مقدار دهی شده است. به دلیل اینکه Z و z فاقد پارامتر خاصی هستند هردوی آنها به طور یکسان عمل خواهند کرد.
L نسبی	(x y) +	lineto	یک خط از نقطه کنونی با مقدار (x, y) رسم می‌کند. در صورتی که با L بزرگ دستور صادر شود مختصات مطلق را به دنبال خواهد داشت. در صورتی که از l کوچک دستور صادر شود مختصات نسبی اعمال خواهد شد.

یک خط عمودی از نقطه کنونی با مقدار (cpx, cpy) رسم می‌کند. در صورتی که با H بزرگ دستور صادر شود مختصات مطلق را به دنبال خواهد داشت. در صورتی که از h کوچک دستور صادر شود مختصات نسبی اعمال خواهد شد.	horizontal lineto	x+	H مطلق h نسبی
یک خط افقی از نقطه کنونی با مقدار (cpx, cpy) رسم می‌کند. در صورتی که با H بزرگ دستور صادر شود مختصات مطلق را به دنبال خواهد داشت. در صورتی که از h کوچک دستور صادر شود مختصات نسبی اعمال خواهد شد.	vertical lineto	y+	V مطلق v نسبی

در صورتی که نیاز داشته باشید در رابطه با انواع دستورات در رابطه با فناوری SVG به لینک : <https://www.w3.org/TR/SVG/paths.html> به لینک مراجعه نمایید.

## PathAnimation نوع

این نوع یکی از مهم‌ترین و پر کاربردترین انواع QML است که برای طراحی مسیرهای سفارشی در انیمیشن بکار می‌رود. نوع PathAnimation این امکان را فراهم می‌کند تا به طور دلخواه مسیری را رسم و مشخص کنیم و بر اساس آن شیء مورد نظر را به حرکت در آوریم. مسلماً کار با این نوع بسیار لذت بخش خواهد بود.

نوع	صفت
point	<b>anchorPoint</b>
int	<b>duration</b>
easing list	<b>easing</b>
real	<b>endRotation</b>
enumeration	<b>orientation</b>
real	<b>orientationEntryDuring</b>
real	<b>orientationExitDuring</b>
Path	<b>path</b>
item	<b>target</b>

- صفت anchorPoint ویژگی لنگر را برای ثابت نگه داشتن یک شیء در محور مسیر را بر عهده دارد.
- صفت duration مقدار زمان بر حسب میلی ثانیه را برای اجرا در مسیر را بر عهده دارد.
- صفت easing لیست خصوصیات نحوه روان سازی را بر روی مسیر بر عهده دارد.
- صفت endRotation وظیفه چرخش برای پایان دادن به مسیر را بر عهده دارد.
- صفت orientation وظیفه کنترل چرخش در طول مسیر را بر عهده دارد.
- صفت orientationEntryDuring مقدار زمان بر اساس میلی ثانیه را برای وارد شدن در جهت مسیر را بر عهده دارد.
- صفت orientationExitDuring مقدار زمان بر اساس میلی ثانیه را برای خارج شدن در جهت مسیر را بر عهده دارد.

- صفت path نگه دارنده مسیر حرکت است.

- صفت target آیتم مورد نظر را برای حرکت بر روی مسیر را بر عهده دارد.

برای درک بهتر ابتدا مثالی را در نظر می‌گیریم که در امتداد یک خط راست شیئی را با نام Box به حرکت در می‌آوریم.

```

Rectangle {
    id: rect
    width: 256
    height: 256
    color: "#fff"
    anchors.centerIn: parent
}

Canvas {
    id: canvas
    anchors.fill: parent
    antialiasing: true

    onPaint: {
        var context = canvas.getContext("2d")
        context.clearRect(0, 0, width, height)
        context.strokeStyle = "green"
        context.path = pathAnim.path
        context.stroke()
    }
}

SequentialAnimation {
    running: true
    loops: -1

    PauseAnimation { duration: 1000 }
    PathAnimation {
        id: pathAnim

        duration: 2000
        easing.type: Easing.InOutExpo

        target: box
        orientation: PathAnimation.Fixed
        anchorPoint: Qt.point(box.width/2, box.height/2)

        path:

            Path {
                id:myPath
                startX: 0; startY: 180

                PathLine {
                    x: 180
                    y: 180
                }
            }

            onChanged: canvas.requestPaint()
    }
}

Rectangle {

```

```

id: box

x: myPath.hstartX - 25; y: myPath.hstartY - 25
width: 50; height: 50
border.width: 1
antialiasing: true
color: "green"
border.color: "darkgreen"

Text {
    anchors.centerIn: parent
    text: "Box"
    color: "white"
}
}

```

ابتدا محیطی را با ابعاد ۲۵۰ در ۲۵۰ پیکسل در نظر می‌گیریم که با نام rect از نوع Rectangle مشخص گردیده است. سپس از نوع Canvas برای ترسیم خطوط مرتبط با مسیر که در صفحات قبل کتاب به آن اشاره شد استفاده می‌کنیم. نوع PathAnimation را با نام pathAnim مشخص کرده و مشخصه‌های easing و duration آن را مشخص می‌کنیم. آیتمی با نام box از نوع Rectangle ساخته و به آن ویژگی‌های رنگ و متن داخل آن توسط نوع Text را اعمال می‌کنیم. حال باید مشخصه target مرتبط به PathAnimation را برابر box قرار دهیم.

صفت orientation را با مقدار PathAnimation.Fixed مشخص می‌کنیم تا در نقطه شروع وضعیت چرخش در هر حالتی ثابت باقی بماند. لازم است صفت anchorPoint را برابر مقادیر شیء box تنظیم کنیم تا آن شیء را در جای خود تنظیم نماید. مشخصه‌های لنگر سازی برابر است با طول و عرض مرتبط با شیء با نام box حال صفت path مشخص می‌کند که در چه مسیری باید حرکت کند، مقدار آن را برابر با یک رسم خط مستقیم با مشخصات نقطه آغاز ۱۸۰ تا ۱۸۰ درجه مشخص کرده‌ایم.

توجه داشته باشید که خود PathLine با نقطه آغاز ۱۸۰ و ۱۸۰ در محور x و y مشخص شده است. رویداد برای canvas onChanged یک رویداد برای است که مشخص می‌کند درخواست برای بوم ترسیم اعمال خواهد شد. این گزینه در صورتی مورد استفاده قرار می‌گیرد که در حین ترسیم تغییرات صورت بگیرد. بنابراین در این مثال تاثیر خاصی نخواهد داشت.



نتیجهً کد مربوطه حرکت یک جعبه در امتداد یک خط مستقیم است.

ممکن است نیاز باشد تا شیء ای را در امتداد خطوط پیچیده یا منحنی به حرکت در آوریم که در این صورت کافی است کدهای مربوط به بخش Path و مورد نیاز را تغییر دهیم که به صورت زیر آمده است:

```

Rectangle {
    id: rect
    width: 500
    height: 500
    color: "#ffff"
}

```

```

Canvas {
    id: canvas
    anchors.fill: parent
    antialiasing: true

    onPaint: {
        var context = canvas.getContext("2d")
        context.clearRect(0, 0, width, height)
        context.strokeStyle = "green"
        context.path = pathAnim.path
        context.stroke()
    }
}

SequentialAnimation {
    running: true
    loops: -1

    PauseAnimation { duration: 1000 }
    PathAnimation {
        id: pathAnim

        duration: 2000
        easing.type: Easing.InOutExpo

        target: box
        orientation: PathAnimation.RightFirst
        anchorPoint: Qt.point(box.width/2, box.height/2)

        path: Path {
            startX: 50; startY: 50

            PathCubic {
                x: rect.width - 50
                y: rect.height - 50

                control1X: x; control1Y: 50
                control2X: 50; control2Y: y
            }

            onChanged: canvas.requestPaint()
        }
    }
}

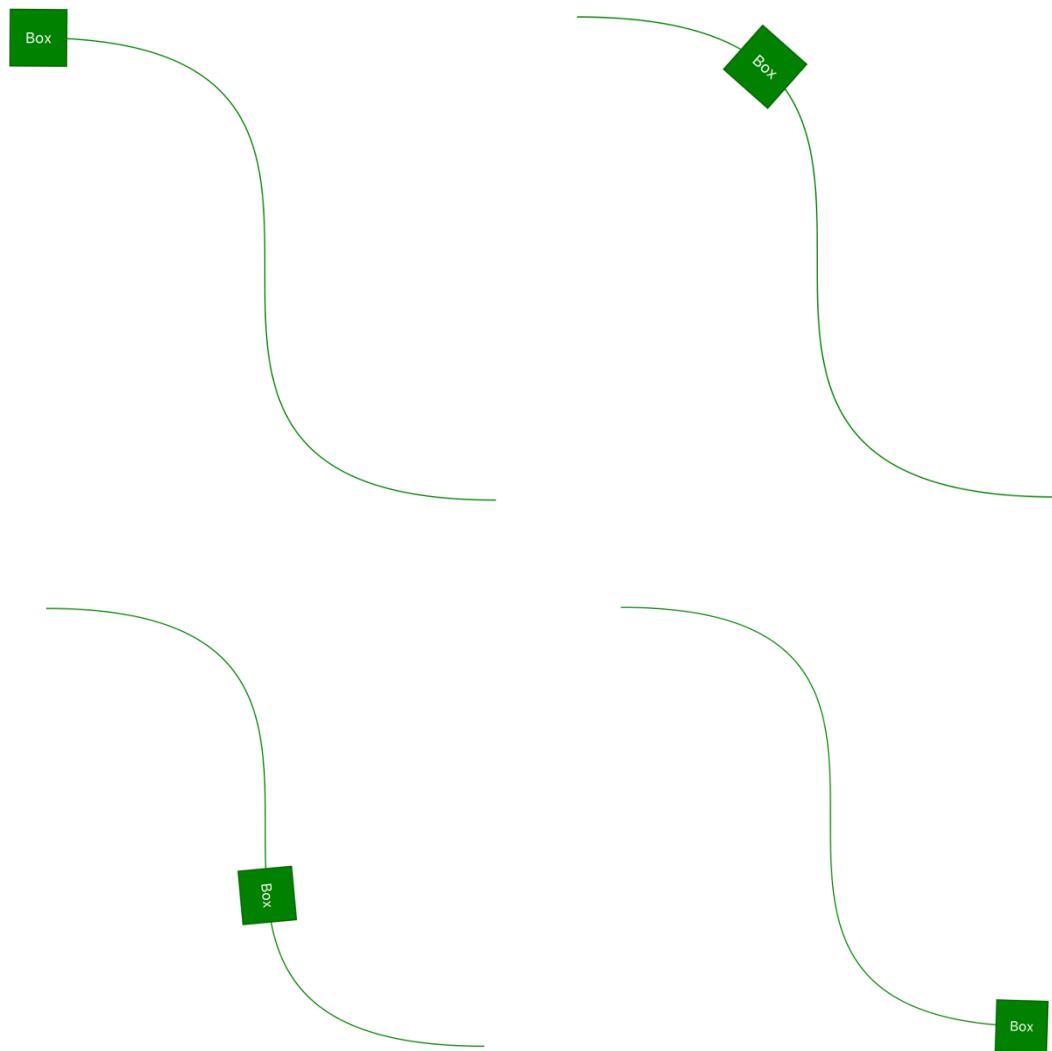
Rectangle {
    id: box

    x: 25; y: 25
    width: 50; height: 50
    border.width: 1
    antialiasing: true
    color: "green"
    border.color: "darkgreen"

    Text {
        anchors.centerIn: parent
        text: "Box"
        color: "white"
    }
}

```

در کد فوق در داخل مشخصه path در نوع Path استفاده شده است که نتیجه نهایی آن به صورت زیر است.



تصاویر فوق نشان‌دهنده مسیر حرکتی بر اساس نوع **Path** ای است که ساخته‌ایم.

شیء مربوطه می‌تواند یک دکمه جهت عملکرد در برنامه یا شیء ای در یک محیط بازی باشد! این به شما بستگی دارد که از خاصیت Path در چه زمینه‌ای استفاده کنید.

مثال دیگر در حالت PathSVG به روش زیر پیاده سازی شده است:

```
Rectangle {  
    id: rect  
    width: 500  
    height: 500  
    color: "#fff"  
  
    Canvas {  
        id: canvas  
        anchors.fill: parent  
        antialiasing: true  
  
        onPaint: {
```

```

        var context = canvas.getContext("2d")
        context.clearRect(0, 0, width, height)
        context.strokeStyle = "green"
        context.path = pathAnim.path
        context.stroke()
    }

}

SequentialAnimation {
    running: true
    loops: -1

    PauseAnimation { duration: 1000 }
    PathAnimation {
        id: pathAnim

        duration: 2000
        easing.type: Easing.InOutExpo

        target: box
        orientation: PathAnimation.RightFirst
        anchorPoint: Qt.point(box.width/2, box.height/2)

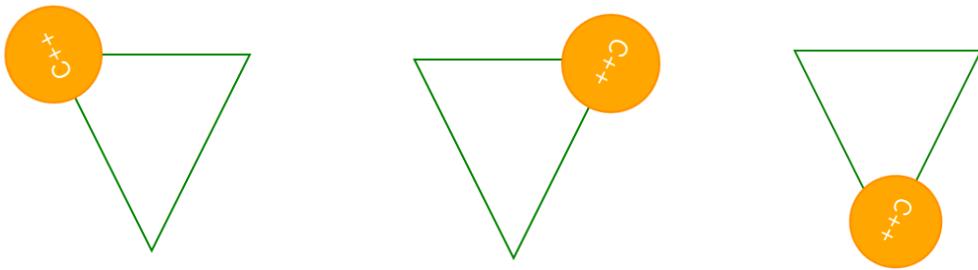
        path: Path {
            startX: 50; startY: 50
            PathSvg { path: "L 150 50 L 100 150 z" }
            onChanged: canvas.requestPaint()
        }
    }
}

Rectangle {
    id: box

    x: 25; y: 25
    width: 50; height: 50
    border.width: 1
    antialiasing: true
    color: "orange"
    border.color: "darkorange"
    radius: 50

    Text {
        anchors.centerIn: parent
        text: "C++"
        color: "white"
    }
}
}

```



## PathView نوع

این نوع امکان ارائه آیتم (اقلام) را بر روی یک مسیر مشخص فراهم می‌کند. در مثال زیر نحوه استفاده از PathView آورده شده است. ابتدا مدلی از ListModel با عناصر مربوطه ایجاد و سپس توسط نوع Path مسیر مشخصی را برای نمایش در PathView ارسال می‌کنیم.

```

ListModel {
    id: model
    ListElement {
        name: "Number 1"
        icon: "qrc:/environment.png"
    }
    ListElement {
        name: "Number 2"
        icon: "qrc:/light-bulb.png"
    }
    ListElement {
        name: "Number 3"
        icon: "qrc:/drop.png"
    }
}

Rectangle {
    width: 240; height: 200
    Component {
        id: delegate
        Column {
            id: wrapper
            Image {
                anchors.horizontalCenter: nameText.horizontalCenter
                width: 64; height: 64
                source: icon
            }
            Text {
                id: nameText
                text: name
                font.pointSize: 9
                color: wrapper.PathView.isCurrentItem ? "red" : "black"
            }
        }
    }
}

PathView {
    anchors.fill: parent
    model: model
    delegate: delegate
    path: Path {
        startX: 120; startY: 100
        PathQuad { x: 120; y: 25; controlX: 260; controlY: 75 }
        PathQuad { x: 120; y: 100; controlX: -20; controlY: 75 }
    }
}

```

```

        }
    }
}

```

در کد فوق PathView دارای مشخصه model برای دریافت مدل مورد نظر است که می‌تواند از طرف C++ نیز ارسال شود. این بستگی به شما دارد که مدل داده‌ای خود را از کجا دریافت خواهید کرد. مشخصه delegate همانطور که قبلاً به آن اشاره شده است نوع آیتم مورد نظر را درخواست می‌کند که در قالب Component یک (مؤلفه) ایجاد شده است.



خروجی که در تصویر فوق مشاهده می‌کنید توسط PathView به نمایش در آمده است که در بستر یک مسیر دو بعدی با گرفتن و حرکت دادن آیتم‌ها به چرخش در می‌آیند. این مثال را می‌توان به یک منوی کاملاً سفارشی تبدیل کرد.

## PauseAnimation نوع

از نوع PauseAnimation زمانی استفاده می‌شود که از SequentialAnimation مورد استفاده قرار گرفته باشد. با استفاده از این نوع می‌توان یک وقفه در اجرای انیمیشن بر اساس زمان مشخص کرد.

```

Rectangle {
    id: rect
    width: 100; height: 100
    color: "green"

    SequentialAnimation {
        running: true
        NumberAnimation { target: rect; property: "x"; to: 50; duration: 1000
    }
        PauseAnimation { duration: 2000 }
        NumberAnimation { target: rect; property: "y"; to: 50; duration: 1000
    }
}
}

```

در کد فوق در بین دو کد SequentialAnimation از نوع PauseAnimation استفاده کرده‌ایم که در بازه زمانی ۲۰۰۰ میلی ثانیه یک وقفه در اجرای انیمیشن ایجاد می‌کند.

## نوع PropertyAction

نوع زمانی مورد استفاده قرار می‌گیرد که نیاز باشد سریعاً در طول یک اینیمیشن یک تغییر اعمال شود. تغییرات اعمال شده متحرک نیستند. استفاده از این نوع در مشخصه‌های غیر متحرک مفید است.

## نوع PropertyChanges

نوع **PropertyChanges** جهت معرفی مقادیر یا اتصالاتی در یک حالت (وضعیت) مورد استفاده قرار می‌گیرد. به طور کلی باعث می‌شود که مقادیر آن در موقع تغییر بین وضعیتی به وضعیتی دیگر عوض شوند. به مثال زیر توجه کنید:

```
Item {
    id: container
    width: 300; height: 300

    Rectangle {
        id: rect
        width: 100; height: 100
        color: "grey"

        MouseArea {
            id: mouseArea
            anchors.fill: parent
        }

        states: State {
            name: "resized"; when: mouseArea.pressed
            PropertyChanges {
                target: rect; color: "green";
                width: container.width;
                height: container.height
            }
        }
    }
}
```

در مثال فوق زمانی که بر روی شیء از قبل تعریف شده با مشخصه `rect` فشار وارد شود رنگ و اندازه آن تغییر خواهد کرد که در اینجا ابعاد طول و عرض آن در زمان تغییر برابر با طول و عرض والد آن یعنی `container` قرار می‌گیرد.

## نوع Rectangle

آیتم مستطیل (Rectangle) جهت پر کردن منطقه‌ای با رنگ جامد و یا یک شیپ در قالب مستطیل را فراهم می‌کند. در نظر داشته باشید یکی از پر مصرف‌ترین آیتم‌های موجود در QML نوع `Rectangle` است که با آن می‌توانید اشیاء مختلفی را جلوه بصری بیخشید. ساختار کلی آن به صورت زیر است:

```

Rectangle {
    width: 100
    height: 100
    color: "#bebebe"
    border.color: "#777"
    border.width: 2
    radius: 10
}

```

نوع	صفت
bool	<b>antialiasing</b>
int	<b>border</b>
color	<b>color</b>
Gradient	<b>gradient</b>
real	<b>radius</b>

نوع مستطیل دارای خصیصه‌های مهمی می‌باشند که در CSS نیز به آن‌ها اشاره شده است. جهت اعمال رنگ در زمینه شیء از خاصیت **color** استفاده می‌کنیم که در صورت نیاز به حالت گرادیان از خاصیت **gradient** استفاده خواهد شد. جهت مقدار دهی به لبه‌های شیء از خاصیت **radius** و برای اجبار در خوش‌نمایشدن از صفت **antialiasing** استفاده خواهیم کرد. جهت درک بهتر این موضوع، کد مذکور مثال تصویر زیر را تولید خواهد کرد:

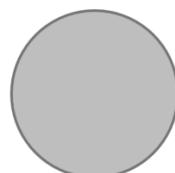


حال فرض کنید نیاز است یک شکل در قالب دایره ایجاد کنیم، در این صورت کافی است خاصیت لبه‌های آن را مقدار دهی کنید. مقدار **radius** باید نصف مقدار طول و عرض یک شیء باشد تا خاصیت گردی آن به حداثتی خود برسد. برای مثال در صورتی که مقادیر طول و عرض یک مستطیل برابر باشند با **radius** ۶۴ در ۱۲۸ و یک حالت مربع را تولید نماید در این صورت مقدار **radius** آن باید برابر با ۶۴ باشد.

```

Rectangle {
    width: 128
    height: 128
    color: "#bebebe"
    border.color: "#777"
    border.width: 2
    radius: 64
}

```



نوع مستطیل بسیار کارآمد است لذا در یک پروژه ممکن است بخش‌های بسیاری از آن را تشکیل دهد. برای مثال یک دکمه کاملاً سفارشی را می‌توانید ایجاد کرد.

```
Rectangle {
```

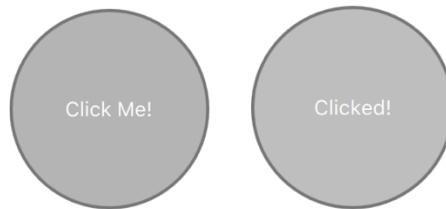
```

width: 128
height: 128
border.color: "#777"
border.width: 2
radius: 64
color: mouseArea.containsPress ? "#bebebe" : "#b4b4b4"

Text {
    text: mouseArea.containsPress ? "Clicked!" : "Click Me!"
    anchors.centerIn: parent
    color: "#fff"
}

MouseArea {
    id:mouseArea
    anchors.fill: parent
    onClicked: {
        console.log("Clicked!");
        //Run your C++ function in here.
    }
}

```



کد فوق توسط یک نوع Rectangle ابتدا یک شیء دایره‌ای را ایجاد و سپس با اختصاص دادن یک نوع MouseArea بر آن و سفارشی سازی به یک دکمه قابل کلیک باز طراحی شده تبدیل شده است. این مثال نشان می‌دهد که در QML شما می‌توانید حتی اشیاء موجود را باز طراحی نمایید و برای محصول خود ساختاری کاملاً سفارشی ایجاد و آن را از روش سنتی جدا نمایید.

## RegExpValidator نوع

نوع امکان اعتبار سنجی برای انواع رشته‌ای را که بر اساس عبارات منظم مشخص می‌شوند را فراهم می‌کند.

نوع	صفت
regExp	<b>regExp</b>

در برنامه‌ها ممکن است جایی نیاز باشد که مقداری را اعتبار سنجی کنید. این بیشتر در بخش رابط‌کاربری مورد نیاز است که مقادیر وارد شده از سمت کاربر مورد ارزیابی قرار گیرند. برای مثال فرض کنید می‌خواهیم فیلد ورودی ما تنها مجاز به دریافت رشته باشد. در این صورت باید مقدار اعتبار سنج شیء ورودی را برابر با regExp که شامل فرمول سفارشی است قرار دهیم. صفت **validator** در انواع **TextField** و **TextInput** در دسترس است.

ساختار آن به این صورت است: **RegExpValidator{regExp: /Your Regex Code/}**

```
TextInput {
```

```

font.pixelSize: 16
width:100
validator: RegExpValidator{regExp: /[a-zA-Z]+/}
}

```

کد فوق نشان می‌دهد که با درج کد `/[a-zA-Z]/` که عمل محدود سازی ورودی در بازه حروف لاتین A تا Z و a تا z را بر عهده دارد بر صفت اعتبار سنج TextInput وارد شده است که نتیجه آن بعد از اجرا تنها مجاز به ورود حروف الفبای لاتین خواهد بود. مثال دیگر در رابطه با این خاصیت جهت ساخت قالب یا پترن (شکل) ورودی آدرس ایمیل آمده است که به صورت زیر خواهد بود:

```

TextField {
    id:textReg
    font.pixelSize: 16
    width:100
    validator: RegExpValidator{
        regExp: /^[a-z0-9_\.-]+@[^\da-zA-Z\.-]+\.[a-zA-Z\.]{2,6}$/,
    }
}

```

جهت سفارشی سازی در این زمینه می‌توانید بر اساس قوانین عبارات منظم اقدام کنید. توجه داشته باشید کدهای عبارت منظم باید بین دو اسلش `/.../` قرار بگیرد. این موارد را می‌توانید بر اساس استاندارد عبارت‌های منظم در اختیار داشته باشید. کافی است از آدرس‌های زیر استفاده کنید: <http://regexpr.com/>, <http://www.regextester.com/> در این آدرس‌ها می‌توانید پترن‌های مورد نظر خود را تولید و بررسی کنید. فراموش نکنید که عبارت‌های منظم خارج از مبحث کتاب است و شما باید خود در رابطه با نحوه تولید آن‌ها تحقیق کنید.

## نوع Repeater

نوع Repeater مناسب برای ساخت آیتم‌های مشابه به تعداد بسیار زیاد است. همانند انواع نمایه دیگر یک Repeater دارای صفات `model` و `delegate` است. نوع تکرار کننده توسط آیتم‌های مستقر ساز دیگری مانند Row و Column احاطه شده است که جهت نمایش داده‌های تکرار شونده توسط Repeater لازم هستند.

نوع	صفت
<code>int</code>	<code>count</code>
<code>Component</code>	<code>delegate</code>
<code>any</code>	<code>model</code>

ساختار این نوع به صورت زیر است که می‌بایست همراه با نوع Row و یا Column برای ساخت حالت بصری ستونی یا ردیفی اعمال شود.

```

Row {
    spacing: 5
    Repeater {
        model: 5
        Rectangle {
            width: 64; height: 64
            border.width: 1
            color: "green"
    }
}

```

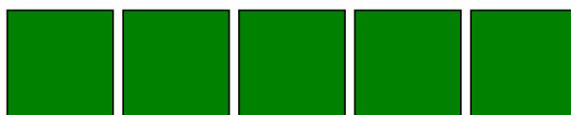
```

        }
    }
}

```

در کد فوق مقدار ۵ به صفت model داده شده است که به معنی آن است که آیتم‌های موجود در تکرار شونده به تعداد ۵ عدد تکرار شوند. این زمانی است که ما مقدار دستی را بدون دریافت مقادیر تابع از سمت C++ و یا جاوااسکریپت می‌سازیم.

کافی است در داخل بدن Repeater شیء مورد نظر خود را اضافه کنید نتیجه نهایی آن به صورت زیر خواهد بود:



با توجه به خاصیت delegate و پشتیبانی از Component ممکن است شیء یا داده‌ای را به طور سفارشی طراحی و آن را برای تکرار کننده ارجاع دهیم. در این صورت به روش زیر اعمال خواهد شد.

```

Row {
    spacing: 5
    Repeater {
        model: 5
        delegate: myComponent
    }
}

```

```

Component {
    id:myComponent
    Rectangle {
        width: 32
        height: 32
        color: "green"
        radius: 16
    }

    Text {
        text: index
        anchors.centerIn: parent
        color: "#fff"
    }
}

```

در کد فوق با ساخت یک کامپوننت با نام myComponent شیء‌ای را با شکل دایره و متن برگرفته از شاخص‌ها (index) فراهم کرده‌ایم سپس در نوع Repeater خاصیت delegate برابر با myComponent قرار گرفته است که یک ویژگی خوب جهت دریافت اطلاعات است.



خاصیت مدل (model) امکان این را فراهم می‌کند که داده‌های سفارشی را در قالب یک مدل پیاده سازی کنیم. توجه داشته باشید که اگر مدلی شامل لیستی از اشیاء و یا رشته‌ها باشد مقادیر آن را می‌توانند در حالت فقط خواندنی دریافت شوند که توسط مشخصه modelData و به صورت رشته و یا شیء نگه داری و مورد استفاده قرار می‌گیرد. کد زیر نمونه‌ای از نحوه استفاده از مشخصه model است.

```

Row {
    spacing: 5
    Repeater {
        model: ["C++", "QML", "JavaScript"]
    }
}

```

```

        delegate: myComponent
    }

}

Component {
    id:myComponent

    Rectangle {
        width: 100
        height: 48
        color: "grey"

        Text {
            text: modelData
            anchors.centerIn: parent
            color: "#fff"
        }
    }
}

```

مشخصه مدل (model) در Repeater مقدیری را دریافت کرده است که برای دسترسی به آنها در قالب رشته از مشخصه modelData در مقدار text از نوع Text {} استفاده شده است.

این موارد بیشتر در فرانت‌لند بکارگرفته می‌شوند، بنابراین در صورتی که نیاز باشد لیستی از داده‌ها را از سمت بکاند مدل سازی کنیم کافی است آن در قالب کلاس و تابع نوع بازگشتی لیست کننده پیاده سازی کنیم. برای مثال کلاسی را با نام MyClass می‌سازیم. فایل سرآیند با پسوند .h. شامل کد زیر خواهد بود:

```

#ifndef MYCLASS_H
#define MYCLASS_H

#include <QObject>
#include <QStringList>

class MyClass : public QObject
{
    Q_OBJECT
public:
    MyClass();
    Q_INVOKABLE QStringList myModel();
};

#endif // MYCLASS_H

```

کد فایل MyClass.h. به صورت زیر پیاده سازی می‌شود:

```

#include "myclass.h"

MyClass::MyClass() { }

QStringList MyClass::myModel () {
    QStringList fonts;

    fonts << "Arial"
        << "Helvetica"

```

```

    << "Times"
    << "Courier";

return fonts;
}

```

همانطور که قبلا اشاره شده است کلاس مربوطه بر پایه **QObject** پیاده سازی شده و سپس تابع مورد نظر از نوع لیست رشته‌ای ایجاد شده است. در بدنه تابع از کلاس **QStringList** مشتق گرفته شده و سپس لیستی از عناوین فونت (نام قلم) در آن ارسال شده است که برای دریافت این اطلاعات از سمت QML ابتدا آن را رجیستر خواهیم کرد.

```

#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QCoreApplication>

#include "myclass.h"

int main(int argc, char *argv[])
{
    QGuiApplication app(argc, argv);

    qmlRegisterType<MyClass>("com.genyleap.book", 1, 0, "MyClass");
    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}

```

با توجه به مفاهیم رجیستر کننده که در کتاب آمده است کلاس فوق را ثبت و در سند QML به صورت زیر از آن استفاده خواهیم کرد:

```

import QtQuick 2.9
import QtQuick.Window 2.2

//Importing My C++ Class
import com.genyleap.book 1.0

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    MyClass { id:myClass; }

    Row {
        spacing: 5
        Repeater {
            model: myClass.myModel(); call your c++ function.
            delegate: myComponent
        }
    }

    Component {
        id:myComponent
    }
}

```

```

Rectangle {
    width: 100
    height: 48
    color: "grey"
}

Text {
    text: modelData
    anchors.centerIn: parent
    color: "#fff"
}

}
}
}

```

طبق دستور فوق ابتدا کلاس مربوطه را وارد و سپس از آن یک مشتق ساخته و تابع را برابر مشخصه model در نوع Repeater قرار می‌دهیم که بعد از اجرا خروجی آن به شکل زیر خواهد بود:

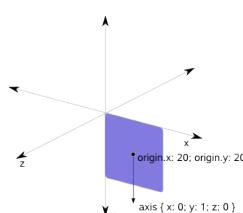


## نوع Rotation

نوع Rotation امکان چرخش یک آیتم را بر اساس چرخش از نوع تبدیل (transform) را فراهم می‌کند. این نوع اجازه می‌دهد تا آیتم در محور Z نسبت به یک نقطه دلخواه به چرخش در آید و همچنین روشی را برای چرخش شبیه سه بعدی برای آیتم‌ها فراهم می‌کند.

نوع	صفت
real	angle
real	axis
real	origin

در صفت axis (محور) برای چرخش حول نقطه. زاویه برای چرخش در جهت عقربه‌های ساعت را فراهم می‌کند. که برای چرخش ساده‌ی دو بعدی اطراف یک نقطه مورد در نظر گرفته شده است، در نظر داشته باشید که شما نیاز به تعیین یک محور نیستید، چون که محور z به عنوان پیش‌فرض محور چرخش است. برای یک چرخش نوعی سه بعدی شما هم مبدأ و هم محور را تعیین خواهید کرد:



همچنین ویژگی origin نقطه مبدأ از چرخش نقطه‌ای را به عنوان یک محور ثابت نگه می‌دارد که به طور پیشفرض مقادیر مبدأ آن (0,0) است. مثال زیر نحوه استفاده از این نوع را نمایش می‌دهد:

```

Rectangle {
    width: 100; height: 100
    color: "green"
    transform: Rotation { origin.x: 35; origin.y: 35; angle: 45}
}

```

}



همانطور که اشاره شد جهت استفاده از خاصیت چرخش در نوع مورد نظر آن را برابر با ویژگی **transform** شیء قرار خواهیم داد. کد فوق چرخش شیء Rectangle را در حول نقاط داخلی ۳۵ و ۳۵ نشان می‌دهد. این حالتی است که به صورت دو بعدی بوده و جهت استفاده از این خاصیت در قالب شبیه به سه-بعدی شما باید محورهای چرخش را علاوه بر نقطه مبدأ آن مشخص کنید. مثال زیر نشان می‌دهد که چگونه یک تصویر در حالت‌های مختلف چرخش داده می‌شود که مانند یک تصویر سه بعدی به نظر خواهد آمد.

```
Row {  
    x: 10; y: 10  
    spacing: 10  
  
    Image { source: "qrc:/temperature.png"; }  
    Image {  
        source: "qrc:/temperature.png"  
        transform: Rotation {  
            origin.x: 30;  
            origin.y: 30;  
            axis { x: 0; y: 1; z: 0 }  
            angle: 18;  
        }  
    }  
    Image {  
        source: "qrc:/temperature.png"  
        transform: Rotation {  
            origin.x: 30;  
            origin.y: 30;  
            axis { x: 0; y: 1; z: 0 }  
            angle: 32;  
        }  
    }  
    Image {  
        source: "qrc:/temperature.png"  
        transform: Rotation {  
            origin.x: 30;  
            origin.y: 30;  
            axis { x: 0; y: 1; z: 0 }  
            angle: 47;  
        }  
    }  
    Image {  
        source: "qrc:/temperature.png"  
        transform: Rotation {  
            origin.x: 30;  
            origin.y: 30;  
            axis { x: 0; y: 1; z: 0 }  
            angle: 72;  
        }  
    }  
}
```

در کد ذکر شده از نوع Image جهت نمایش یک تصویر استفاده شده است، با مقدار دهی مشخصه transform آن با rotation مشخصه های angle axis و origin مربوط به آن را مقدار دهی و به ترتیب نمایش داده شده است. که نتیجه آن به صورت زیر خواهد بود.



## نوع RotationAnimation

نوع `PropertyAnimation` یک `RotationAnimation` اختصاصی است که کنترل چرخش در طول یک انیمیشن را در اختیار قرار می دهد. در این نوع به طور پیشفرض، چرخش بر اساس تغییرات در اعداد صورت می گیرد. برای مثال چرخش از  $0^\circ$  تا  $240^\circ$  چرخشی را در محور  $240^\circ$  درجه جهت عقربه های ساعت فراهم می کند. در حالی که یک چرخش از  $240^\circ$  تا  $0^\circ$  درجه چرخشی در بازه  $240^\circ$  درجه بر خلاف عقربه های ساعت ایجاد می کند. به طور کلی این ویژگی می توانید جهتی را مشخص کند که چرخش در آن رخ دهد.

نوع	صفت
enumeration	<code>direction</code>
real	<code>from</code>
real	<code>to</code>

```
Item {
    width: 300; height: 300

    Rectangle {
        id: rect
        width: 150; height: 150; anchors.centerIn: parent
        color: "green"
        antialiasing: true

        states: State {
            name: "rotated"
            PropertyChanges { target: rect; rotation: 180 }
        }

        transitions: Transition {
            RotationAnimation {
                duration: 500;
                direction: RotationAnimation.Clockwise;
            }
        }
    }

    MouseArea { anchors.fill: parent; onClicked: rect.state = "rotated" }
}
```

کد فوق نشان می دهد که با ترکیب `RotationAnimation` در یک حالت چرخشی مربوط به شیء با مشخص کردن خاصیت `direction` آن جهش چرخشی بر اساس مقدار معین شده یعنی جهت عقربه های ساعت صورت خواهد گرفت. در صورتی که `RotationAnimation` برابر با `Clockwise` باشد شیء مورد نظر بر خلاف جهت عقربه های ساعت به چرخش در خواهد آمد.

## RotationAnimator نوع

نوع Animation با انواع Animator ها فرق می‌کند. هنگامی که از یک Animator استفاده شود، انیمیشن مربوطه می‌تواند در نخ ایجاد شده اجرا و ارزش - مقادیر مشخصه‌های آن مانند from یا to در زمان به پایان رسیدن انیمیشن رو به پایان پرش یابند. برای مثال فرض کنید مربعی را به حالت چرخش در آورده‌اید و این چرخش قرار است در زمان اتمام یک انیمیشن کنترل و به پایان برسد. در این صورت بهتر است از نوع استفاده شود که در کد زیر مثالی از آن آورده شده است:

```
Rectangle {
    id: rotatingBox
    width: 50
    height: 50
    color: "green"
    RotationAnimator {
        target: rotatingBox;
        from: 0;
        to: 360;
        duration: 200
        running: true
    }
}
```

این امکان وجود دارد که کنترل انیمیشن را توسط کلمه‌کلیدی on در اختیار داشته باشیم:

```
Rectangle {
    width: 100
    height: 100
    color: "green"
    RotationAnimator on rotation {
        from: 0;
        to: 360;
        duration: 1000
    }
}
```

## Row نوع

نوع Row یا همان ردیف از انواع مرتبط با لایه‌های موقعیت می‌باشند که موجب می‌گردد آیتم یا اشیاء مورد نظر در امتداد یک ردیف به نماش در آیند. این نوع باعث می‌شود بدون استفاده از خاصیت‌های لنگر انداختن (anchors) آیتم‌های مورد نظر در موقعیت افقی نمایان شوند. برای مثال کد زیر نشان می‌دهد که چگونه آیتم‌های مختلف در قالب شیء از نوع Rectangle با ابعاد مختلف در یک ردیف افقی نمایش داده می‌شوند.

```
Row {
    spacing: 2
    Rectangle { color: "red"; width: 100; height: 100 }
    Rectangle { color: "green"; width: 35; height: 50 }
    Rectangle { color: "blue"; width: 75; height: 35 }
}
```



## نوع Scale

نوع Scale یا همان مقیاس روشی را برای تغییر حالت دادن به ورش مقیاس پذیری را فراهم می‌کند. این نوع اجازه می‌دهد تا امکان مقیاس پذیری با تغییر مقادیر محورهای x و y صورت گیرد. همچنین اجازه می‌دهد تا این تغییر بر اساس یک نقطه دلخواه انجام شود.

نوع	صفت
real	origin
real	xScale
real	yScale

طبق خاصیت‌های origin و xScale و yScale روش استفاده از این نوع به صورت زیر است که اجازه می‌دهد شکل دایروی ما تغییر حالت دهد.

```
Rectangle {  
    width: 100; height: 100  
    color: "green"  
    radius: 50  
    transform: Scale { origin.x: 50; origin.y: 50; xScale: 2 }  
}
```



## نوع ScaleAnimator

نوع ScaleAnimator همان ویژگی‌های Animator را دارد انواع انیماتورها نوع خاصی از انیمیشن هستند که به طور مستقیم بر روی نمودارهای صحنه‌ای تحت Qt Quick عمل می‌کنند. این در حالتی است که حتی در زمانی که یک پروسه جهت صحنه‌های انیمیشنی در حال اجرا قرار بگیرد انیمیشن مورد نظر تحت این نوع می‌تواند وارد عمل شود. البته بخشی از این قابلیت موازی ساز است که تحت این نوع می‌تواند اعمال شود که مثالی از آن در صفحات قبل در بخش Animator آورده شده است.

## نوع SequentialAnimation

همانطور که در مورد ParallelAnimation قبلاً توضیح داده شده است در اینجا SequentialAnimation اجازه اجرای انیمیشن‌های متعدد را می‌دهد که یکی پس از دیگری اجرا خواهد شد و تنها فرق آن با ParallelAnimation در این است که آن به صورت همزمان انیمیشن‌ها را مورد اجرا قرار می‌دهد.

```

Rectangle {
    id: rect
    width: 50; height: 50
    color: "green"

    SequentialAnimation {
        running: true
        NumberAnimation { target: rect; property: "x"; to: 50; duration:
    1000 }
        NumberAnimation { target: rect; property: "y"; to: 50; duration: 1000
    }
}
}
}

```

کد فوق یک شکل مربع را بر پایه نوع `Rectangle` در محور `x` و `y` به حرمت در می آورد که این امکان توسط `SequentialAnimation` صورت می گیرد. یعنی ابتدا حرکت بر روی محور `x` و سپس بعد از به پایان رسیدن `NumberAnimation` اول حرکت در امتداد محور `y` آغاز می شود.

## نوع `ShaderEffect`

نوع `ShaderEffect` امکان سایه زنی را در راس و قطعه مشخصی (پیکسل) برای نوع مستطیل یا همان `Rectangle` فراهم می کند. این نوع اجزا می دهد تا شما اثراتی را مانند سایه زنی، محو کردن، رنگ آمیزی و ... را توسط QML انجام دهید. البته توجه داشته باشید که نوع سایه زنی به موتور `Qt Quick` در `بک‌اِند` وابسته است بنابراین ممکن است نوع `ShaderEffect` توسط آپشن نرم افزاری در بک اند پشتیبانی نشود و مجبور شوید تا از زبانهای سایه زنی مختلفی با قواعد و انتظارات مختلفی از طرف موتورهای `OpenGL` و `GLSL` استفاده کنید.

برای اینکه مثالی از کاربرد این نوع داشته باشیم در نظر داریم یک شیء با طرح پرچم ایجاد کرده و آن را توسط `ShaderEffect` متحرک سازیم به گونه ای که این پرچم در معرض باد قرار گرفته باشد.

برای اینکار نیاز است از خاصیت `fragmentShader` و یک نوع دیگری با نام `ShaderEffectSource` استفاده کنیم. کد مربوطه به صورت زیر خواهد بود که در آن `sourceItem` خود صفتی برای دریافت شیء مورد نظر شما خواهد بود که در اینجا تصویر پرچم ایران است:

```

Row {
    ShaderEffectSource {
        id: myItem

        sourceItem: Image {
            id: image
            sourceSize.width: 256
            sourceSize.height: 256
            width: 256
            height: 256
            source: "qrc:/iran.png"
        }
    }

    ShaderEffect {
        width: 300
        height: 300
        property var source: myItem
        property real frequency: 8
    }
}

```

```

property real amplitude: 0.05
property real time

NumberAnimation on time {
    from: 0
    to: Math.PI * 2
    duration: 1000
    loops: Animation.Infinite
}

fragmentShader:
"
varying highp vec2 qt_TexCoord0;
    uniform sampler2D source;
    uniform lowp float qt_Opacity;
    uniform highp float frequency;
    uniform highp float amplitude;
    uniform highp float time;
void main() {
    highp vec2 pulse = sin(time - frequency * qt_TexCoord0);
    highp vec2 coord = qt_TexCoord0 + amplitude * vec2(pulse.x, -
pulse.x);
    gl_FragColor = texture2D(source, coord) * qt_Opacity;
}
"
}
}

```

در ادامه کد فوق با بکار گیری نوع `ShaderEffect` و مقدار دهی دستورات رابطه‌ای گرافیکی مانند `OpenGL` در صفت `fragmentShader` عمل متحرک سازی تصویر مربوطه را انجام می‌دهد که در رابطه با کدهای مرتبط با دستورات `OpenGL` مسلماً باید با آنها آشنا باشید که خارج از موضوع و محتوای اصلی این کتاب است.

در ادامه نتیجه کد فوق به صورت زیر خواهد بود، با مشخصه `Animation.Infinite` موجود در `NumberAnimation` به طور بی‌نهایت متحرک خواهد شد.



## ShaderEffectSource نوع

نوع `ShaderEffectSource` امکان ارائه و نمایش آیتم را به عنوان یک بافت فراهم می‌کند.

```

Rectangle {
    width: 200
    height: 100
}
Row {
    opacity: 0.5
}
Item {
    id: myItem
    width: 100; height: 100
    Rectangle { x: 5; y: 5; width: 60; height: 60; color: "#006aff" }
    Rectangle { x: 20; y: 20; width: 60; height: 60; color: "#0058d3" }
}

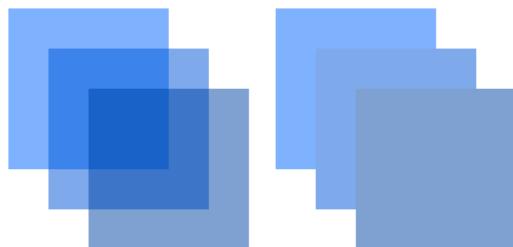
```

```

        Rectangle { x: 35; y: 35; width: 60; height: 60; color: "#0045a0" }
    }
    ShaderEffectSource {
        width: 100; height: 100
        sourceItem: myItem
    }
}

```

کد فوق نشان می‌دهد که شکل تولید شده به عنوان یک بافت ظاهر شده است که مات (بدون شفافیت) است.



## Shortcut نوع

نوع امکان فراهم کردن کلیدهای میانبر را به صورت دستی فراهم می‌کند. به طور کلی میانبرها می‌توانند به عنوان کلیدهای استاندارد تعریف شوند و یا به طور تعریف رشته‌ای در صفت sequence اعمال شوند.

نوع	صفت
bool	autoRepeat
enumeration	context
bool	enable
string	nativeText
string	portableText
keysequence	sequence
list<keysequence>	sequences

نوع کلیدهای میانبر را به صورت دستی فراهم می‌کند. به طور کلی میانبرها می‌توانند به عنوان کلیدهای استاندارد تعریف شوند و یا به طور رشته‌ای در صفت sequence اعمال شوند.

```

Item {

    Shortcut {
        sequence: "Shift+Q"
        onActivated: rect.blue()
        context: Qt.ApplicationShortcut
    }

    Shortcut {
        sequence: "Alt+W"
        onActivated: rect.red()
        context: Qt.ApplicationShortcut
    }

    Item {

```

```

anchors.fill: parent

Rectangle{
    id: rect
    width: 100
    height: 100
    radius: 50
    color: "black"

    function blue() {
        color = "green"
    }
    function red() {
        color = "yellow"
    }
}

}

```

کد مربوطه با اعمال Shortcut های مورد نظر رنگ شیء Rectangle را با فشرده شدن کلیدهای Shift+Q و Alt+W تغییر می‌دهد. خاصیت sequence امکان تعریف کلیدهای ترکیبی را در قالب رشته می‌دهد و خاصیت onActivated زمانی عمل می‌کند که نتیجه کلیدهای فشرده شده صحیح باشند و صفت context بر پایه شمارنده موجود در C++ با نام ShortcutContext عمل می‌کند که به صورت زیر آمده است.

توضیحات	ارزش	ثبت‌ها
این میانبر زمانی فعال است که بر روی والد آن به عنوان فکوس (تمرکز) شده باشد.	.	Qt::WidgetShortcut
این میانبر زمانی فعال است که تمرکز (فکوس) بر روی والد آن به عنوان Widget و یا هر نوع فرزند دیگر صورت بگیرد. این مورد در رابطه با پنجره‌های نوع سوم pop-up ها مورد تاثیر قرار نمی‌گیرند.	۳	Qt::WidgetWithChildrenShortcut
این میانبر زمانی فعال است که والد آن به عنوان Widget به عنوان زیر ویجت‌ها به صورت منطقی تمرکز (فکوس) کرده باشد.	۱	Qt::WindowShortcut
این میانبر زمانی فعال است که پنجره‌های اصلی برنامه فعال باشند.	۲	Qt::ApplicationShortcut

## SmoothedAnimation نوع

نوع SmoothedAnimation مقادیر را بر اساس مقادیر مشخص شده در هدف بر اساس روان‌ساز `ease /in/out` متحرک می‌سازد. فرض کنید قرار است بین دو هدف از اشیاء حرکتی ایجاد شود که ممکن است به صورت بصری مشکلاتی داشته باشد اما با استفاده از این نوع یک حالت صاف و روان بین آن‌ها اتفاق می‌افتد که در ساخت بازی و یا جلوه‌های بصری نرم‌افزارها بسیار کارآمد است.

نوع	صفت
int	<b>duration</b>
int	<b>maximumEasingTime</b>
enumeration	<b>reservingMode</b>
real	<b>velocity</b>

در ادامه مثالی آورده شده است که نشان می‌دهد با فشرده شدن کلیدهای جهشی اشیاء به دنبال همیگر توسط SmoothedAnimation روان سازی شده‌اند.

```
Rectangle {  
    width: 800; height: 600  
  
    Rectangle {  
        width: 60; height: 60  
        x: rect1.x - 5; y: rect1.y - 5  
        color: "gray"  
        Behavior on x { SmoothedAnimation { velocity: 300 } }  
        Behavior on y { SmoothedAnimation { velocity: 300 } }  
    }  
  
    Rectangle {  
        id: rect1  
        width: 50; height: 50  
        color: "orange"  
    }  
  
    focus: true  
    Keys.onRightPressed: rect1.x = rect1.x + 100  
    Keys.onLeftPressed: rect1.x = rect1.x - 100  
    Keys.onUpPressed: rect1.y = rect1.y - 100  
    Keys.onDownPressed: rect1.y = rect1.y + 100  
}
```

در کد فوق مختصات شیء Rectangle کنترل می‌شوند که با مقداردهی به صفت velocity شدت تحرک آن به دنبال فشرده شدن کلیدهای جهشی اعمال می‌شود. اگر نیاز است حالت روان کننده بیشتری به حرکت بین اشیاء اعمال شود بهتر است از صفت Keys.onRightPressed و Keys.onLeftPressed و Keys.onUpPressed و Keys.onDownPressed استفاده شود و مقدار آن بیشتر از مقدار maximumEasingTime وارد شود.

## نوع SpringAnimation

نوع SprintAnimation اینیمیشن فنر، رفتار نوسان‌دار، یک فنر را همراه با ثابت فنر مناسب نمایش می‌دهد تا شتاب و میرایی را طوری تنظیم کند که از بین رفتن اثر نوسانی فنر را کنترل کند. همچنین شما می‌توانید سرعت نهایی اینیمیشن را کنترل کنید.

نوع	صفت
real	damping
real	epsilon
real	mass
real	modulus
real	spring
real	velocity

```
Item {  
    width: 300; height: 300  
    Rectangle {  
        id: rect  
        width: 50; height: 50  
        color: "red"
```

```

        Behavior on x { SpringAnimation { spring: 2; damping: 0.1 } }
        Behavior on y { SpringAnimation { spring: 2; damping: 0.1 } }
    }
    MouseArea {
        anchors.fill: parent
        onClicked: {
            rect.x = mouse.x - rect.width/2
            rect.y = mouse.y - rect.height/2
        }
    }
}

```

مستطیل مورد نظر در زمانی که ماوس کلیک کند با استفاده از `SpringAnimation` به نقطه‌ای که ماوس قرار دارد حرکت می‌کند. استفاده از رفتار نوسانی در راستاهای محورهای `x` و `y` اشاره به این دارد مدامی که این متغیرها تغییر یابند، یک `SpringAnimation` باید اعمال شود. در صورتی که خاصیت `damping` آن را بیشتر کنید حالت فنری حرکت به دنبال آن تشید خواهد شد.

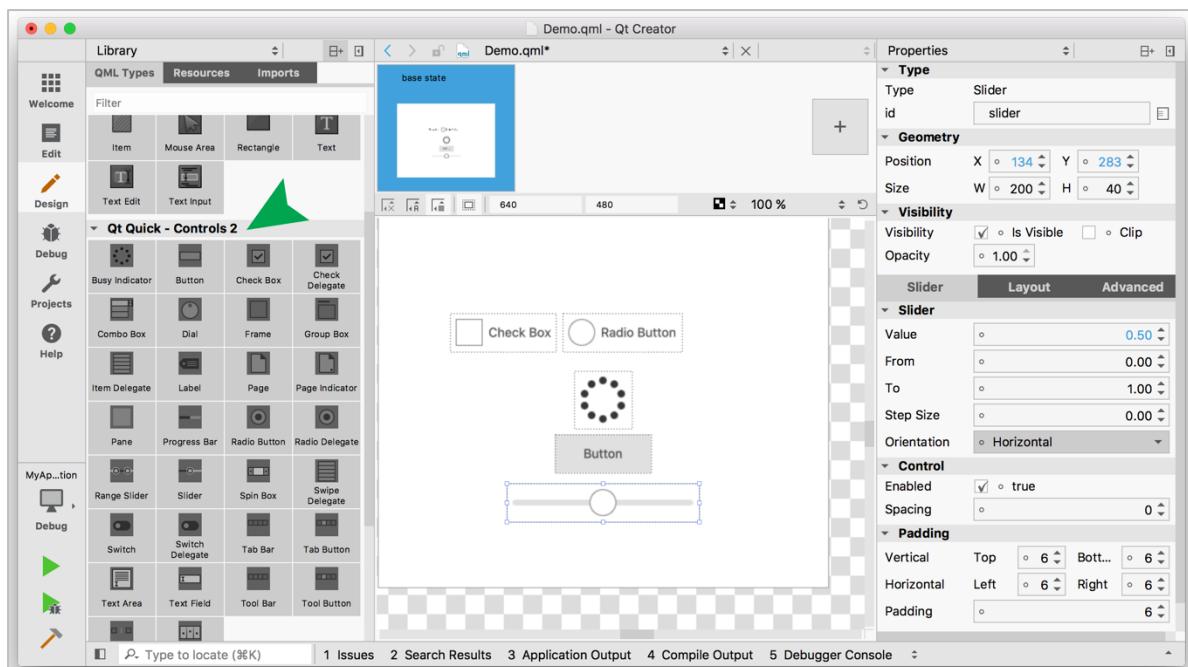
## فصل پنجم

### معرفی انواع کنترل‌های مازولی Qt Quick Controls 2

فناوری کوئیک 2 (Qt Quick) انواع QML را برای ساختن طراحی رابط‌کاربری ارائه می‌کند. کنترل‌های QML، پیوسته با Qt Quick کار می‌کنند. فناوری کوئیک کنترل (Qt Quick Controls 2) در نوع QML می‌تواند در اپلیکیشن شما وارد شده و مورد استفاده قرار گیرند که تحت دستور زیر استفاده از آن ممکن خواهد بود.

```
import QtQuick.Controls 2.2
```

با درج این دستور کنترل‌های مربوط به آن به صورت زیر در دسترس قرار خواهد گرفت.



در زیر جدولی از لیست انواع کنترل‌های موجود در فناوری Qt آمده است که توضیحات و وظایف هر یک از آن‌ها ذکر شده است.

نوع پایه از تمامی کنترل‌ها.	AbstractButton
یک پنجره برنامه‌ای سطح بالا را فراهم می‌کند.	ApplicationWindow

فعالیت‌های در حال بارگذاری را نشان می‌دهد.	<b>BusyIndicator</b>
یک دکمه که می‌تواند توسط کاربر کلیک - فشرده شود.	<b>Button</b>
یک گروه منحصر بفرد دو جانبه از کنترل‌های قابل انتخاب.	<b>ButtonGroup</b>
یک دکمه ویژه‌ای که می‌تواند انتخاب شود و یا در حالت انتخاب نشده قرار گیرد.	<b>CheckBox</b>
یک آیتم محول کننده‌ای است که می‌تواند انتخاب شود و یا در حالت انتخاب نشده قرار گیرد.	<b>CheckDelegate</b>
یک کنترل همراه با لیست و دکمه ترکیب شده که در کمترین فضای ممکن تعییه شده است. همان کنترل لیست کشوئی.	<b>ComboBox</b>
یک ظرف - دربر دارنده از نوع کنترل.	<b>Container</b>
نوع پایه از کنترل‌های رابط‌کاربری.	<b>Control</b>
یک نوع کنترل جهت نمایش پنجره گفتگو.	<b>Dialog</b>
یک کنترل به عنوان شیء فرزند جهت استفاده در کنترل Dialog	<b>DialogButtonBox</b>
یک کنترل چرخشی برای تنظیم یک مقدار، نوعی کنترل شبیه به نوار چرخشی است.	<b>Dial</b>
یک پنل در موقعیت کنار-بغل بر پایه مدل کشوئی (Swipe) را فراهم می‌کند.	<b>Drawer</b>
یک گروه منطقی از کنترل‌ها که در داخل یک قاب تصویری را فراهم می‌کند.	<b>Frame</b>
یک قاب با یک گروه منطقی از کنترل‌ها را فراهم می‌کند.	<b>GroupBox</b>
یک آیتم جهت نمایش آیتمی که می‌تواند به عنوان نمایه‌های مختلف و کنترل‌ها مورد استفاده قرار گیرد.	<b>ItemDelegate</b>
یک برچسب متن با سبک‌های به ارث برده شده و فونت.	<b>Label</b>
کنترل منو که می‌تواند به عنوان یک منوی زمینه و یا منوی پنجره‌ای مورد استفاده قرار بگیرد.	<b>Menu</b>
یک آیتم از منو در داخل منوی اصلی را فراهم می‌کند.	<b>MenuItem</b>
یک کنترل که باعث می‌شود به راحتی سربرگ و پاورقی را به یک صفحه اضافه شود.	<b>Page</b>
صفحه فعل کنونی را نمایش می‌دهد.	<b>PageIndicator</b>
یک پس زمینه مطابق با سبک و پوسته اپلیکیشن را فراهم می‌کند.	<b>Pane</b>
یک نوع پایه پنجره‌ای (Popup) از کنترل‌های رابط‌کاربری است.	<b>Popup</b>
پیشرفت یک عملیات را نمایش می‌دهد که همان نوار وضعیت-نوار پیشرفت است.	<b>ProgressBar</b>
یک دکمه ویژه که می‌تواند با جا به جا شدن روشن یا خاموش شود.	<b>RadioButton</b>
یک آیتم که با محول شدن می‌تواند انتخاب شود و یا در حالت انتخاب نشده قرار بگیرد.	<b>RadioDelegate</b>
یک کنترل لغزنه که برای انتخاب بازه‌ای از مقادیر استفاده می‌شود.	<b>RangeSlider</b>

یک کنترل تعاملی اسکرول بار	ScrollBar
یک کنترل غیر تعاملی اسکرول بار	ScrollIndicator
کنترلی که با انتخاب و کشیدن آن در امتداد یک مسیر مقداردهی می‌شود.	Slider
کنترل جعبه انتخابی - چرخشی (Spin-Box) کنترلی است که اجزه انتخاب یک مقدار از قبل تعیین شده را فراهم می‌سازد.	SpinBox
یک مدل ناوبری - جهت دار را مبتنی بر پشته فراهم می‌کند.	StackView
یک کنترل آیتم با قابلیت محول شدن قابل کشیدن.	SwipeDelegate
امکان کشیدن و حرکت دادن صفحات و همچنین سوئیچ بین آنها را برای کاربر فراهم می‌سازد.	SwipeView
یک دکمه ویژه که می‌تواند با جابجایی وضعیتش خاموش یا روشن شود.	Switch
یک آیتم با قابلیت محول شدن که می‌تواند با جابجایی وضعیتش خاموش یا روشن شود.	SwitchDelegate
یک نوار که می‌تواند شامل با آیکون‌ها باشد و اجزه می‌دهد تا با سوئیچ بین نمایه‌ها و زیر وظایف عملیات مختلفی صورت گیرد.	TabBar
یک کنترل برگه (Tab) که می‌تواند در نوار زبانه مورد استفاده قرار گیرد.	TabButton
یک کنترل ورودی متن با قابلیت چند خطی.	TextArea
یک کنترل ورودی متن با قابلیت یک خطی.	TextField
یک ظرف (نگه دارنده) با کنترل‌های حساس به محیط.	ToolBar
یک دکمه با یک طرح مناسب برای یک نوار ابزار (Tool Bar).	ToolButton
کنترلی که نکات راهنمایی را برای هر نوع کنترلی فراهم می‌کند.	ToolTip
یک کنترل چرخشی - چرخ دنده مانند که با چرخ دادن به آن آیتم‌ها را می‌توان به حالت انتخاب در آورد.	Tumbler

قبل از استفاده از کنترل‌های موجود در QML می‌بایست توجه داشته باشید که برخلاف خصیصه‌ها نام کنترل‌ها باید با کلاکتر بزرگ مشخص شود. برای مثال `Button` صحیح است و `button` نادرست.

کنترل `AbstractButton` نوع پایه‌ای از تمامی کنترل‌ها محسوب می‌شود.

نوع	صفت
bool	<code>autoExclusive</code>
bool	<code>checked</code>
bool	<code>down</code>
Item	<code>indicator</code>
bool	<code>pressed</code>
string	<code>text</code>

توضیح عملکرد سیگنال	سیگنال
زمانی که وضعیت فشردن کلیک ماوس بر روی دکمه از دست می رود و یا زمانی که وضعیت ماوس را بر روی کنترل رها می کنید این سیگنال ساطع خواهد شد.	<b>canceled</b>
زمانی که کاربر بر روی دکمه کلیک می کند این سیگنال ساطع می شود.	<b>clicked</b>
زمانی که کاربر دو بار پشت سر هم بر روی دکمه کلیک کند این سیگنال ساطع خواهد شد.	<b>doubleClicked</b>
زمانی که کاربر بر روی دکمه کلیک کرده و نگه می دارد این سیگنال ساطع می شود.	<b>pressAndHold</b>
با فشرده شدن دکمه توسط کاربر این سیگنال ساطع می شود.	<b>pressed</b>
با رها شدن دکمه توسط کاربر این سیگنال ساطع می شود.	<b>released</b>
توضیح عملکرد تابع	عملکرد
وضعیت مرتبط با دکمه را بررسی می کند.	<b>toggle</b>

کنترلی ایجاد پنجره اصلی برنامه است که در صفحات قبلی در اولین پروژه به نحوه کارکرد آن اشاره شد.

صفت	نوع
<b>activeFocusControl</b>	Control
<b>background</b>	Item
<b>contentData</b>	list<Object>
<b>contentItem</b>	Item
<b>font</b>	font
<b>footer</b>	Item
<b>header</b>	Item
<b>overlay</b>	item
<b>overlay.model</b>	Component
<b>overlay.modeless</b>	Component

کنترل یک کنترل بسیار کار آمد است که و روش استفاده از آن به صورت زیر است :

صفت	نوع
<b>running</b>	bool

```
BusyIndicator {
    running: image.status === Image.Loading
}
```

با در نظر گرفتن خاصیت **running** می توان وضعیت آن را بر اساس شرط موجود تغییر داد که بر اساس نوع بازگشتی **true** و یا **false** مشخص می شود.

## کنترل Button

کنترل **Button** شاید یکی از پر کاربرد ترین کنترل های موجود در طراحی باشد و همانطور که از نامش مشخص است این کنترل قابلیت کلیک شدن را دارد و می تواند رویدادهایی را بپذیرد. روش استفاده از آن به صورت زیر است:

صفت	نوع
<b>autoRepeat</b>	bool
<b>checkable</b>	bool
<b>flat</b>	bool
<b>highlighted</b>	bool

```
Button {
    text: "ثبت اطلاعات"
    onClicked: data.register() // تابع فراخوانی شده از سمت سی پلاس پلاس
}
```

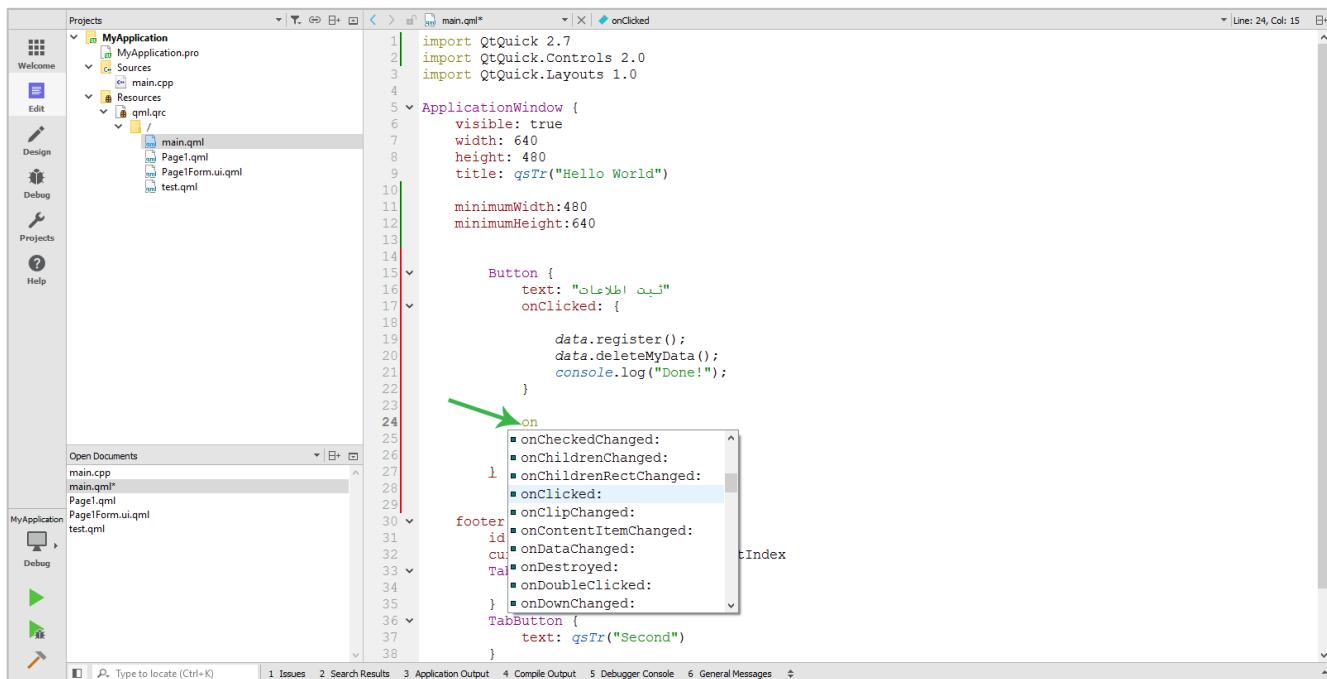
توجه داشته باشید که کنترل ها دارای بلوک شروع و پایان هستند و تمامی زیر عضوها و خصیصه های آن در داخل بلوک ها قابل دسترسی خواهند بود. البته با تعریف شناسه اختصاصی برای هر یک می توان آنها را نیز برای دسترسی و تاثیر پذیری سفارشی سازی کرد. رویدادهای کنترل **Button** شامل **onClicked** است که در صورت وارد نکردن بلوک های آغاز و پایان تنها قادر به پذیرش یک عمل خواهد بود بنابراین جهت استفاده از چند عمل می بایست به صورت زیر اصلاح گردد.

```
Button {
    text: "ثبت اطلاعات"
    onClicked: {
        data.register();
        data.deleteMyData();
        console.log("Done!");
    }
}
```

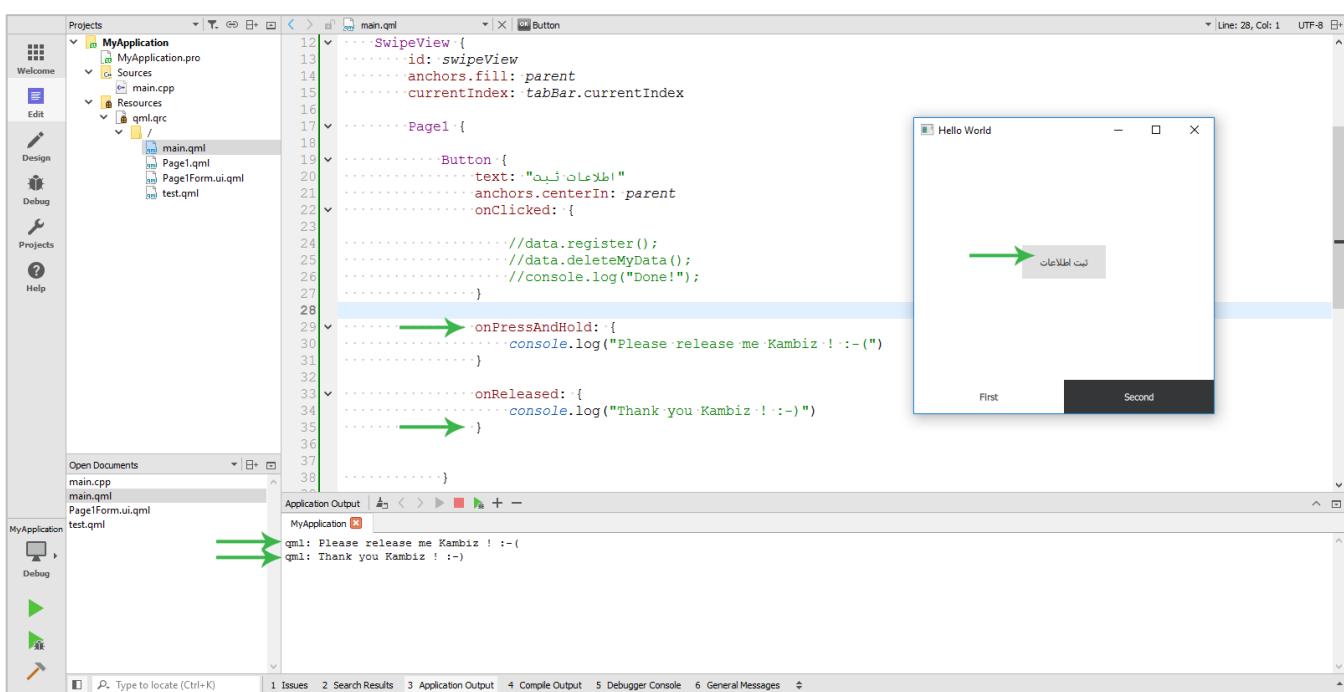
همانطور که مشخص است رویداد **onClicked** با داشتن بلوک می تواند شامل بی نهایت عملیات باشد که از طرف C++ ارسال می گردد. همچنین رویدادهای بسیاری در این کنترل موجود است که می توان به آنها دسترسی پیدا کرد.

جهت دسترسی به رویدادهای مورد نظر می‌توانید با وارد کردن گزینه ... on به لیست رویدادهای قابل پشتیبانی در هر یک از کنترل‌های Qt Quick بروید.

برای مثال در همین کنترل **Button** به صورت زیر لیست رویدادها قابل دسترسی است.



برای درک بهتر مطلب، مثالی را در نظر گرفته‌ایم که در صورت کلیک و نگهداشتن بر روی دکمه، پیغامی را نمایش می‌دهد و در صورت رها کردن آن پیغام آن تغییر می‌کند. بنابراین دو رویداد **onReleased** و **onPressAndHold** گزینه‌های مناسبی برای استفاده از آن است.



با کلیک بر روی دکمه "ثبت اطلاعات" و نگهداشتن آن، ابتدا پیغام "Please release me Kambiz! 😠" نمایش یافته و سپس با رها سازی دکمه پیغام "Thank you Kambiz! 😊" در کنسول محیط توسعه چاپ می‌شود. با استفاده از خاصیت `console.log` عمل چاپ در خروجی صورت

گرفته است. توجه شود که اجرای یک تابع یا حتی چاپ چنین متن ساده‌ای، مستلزم انتخاب صحیح و نوشتن کد در بین بلوک {آکولاڈ} مربوط به رویداد کنترل است. درصورتی که دستورات شما بیش از یک عمل باشد، تنها در بین بلوک‌های مرتبط با رویداد قابل اجرا خواهد بود.

## کنترل **ButtonGroup**

ابزارهای متنوعی در گروهبندی و مرتب‌سازی اشیاء بر روی فرم وجود دارد، در این میان کنترلی با هدف دسته‌بندی و مرتب‌سازی کنترل‌های **Button** موجود است که می‌توان توسط آن انواع کنترل‌ها را علاوه بر گروه بندی کرد. روش استفاده به صورت زیر است:

نوع	صفت
list<AbstractButton>	<b>buttons</b>
AbstractButton	<b>checkedButton</b>
توضیح عملکرد سیگنال	سیگنال
زمانی که کاربر بر روی دکمه کلیک می‌کند این سیگنال ساطع می‌شود.	<b>clicked</b>
توضیح عملکرد تابع	عملکرد
اضافه کردن یک آیتم به صورت دستی در یک گروه از دکمه ضروری نیست. اعضای ضمیمه شده در گروه کنترلی تحت روش ساده‌تری فراهم می‌شوند که تحت این خاصیت صورت می‌گیرد.	<b> addButton</b>
همانند مورد قبلی حذف کردن یک آیتم به صورت دستی در یک گروه از دکمه ضروری نیست. اعضای ضمیمه شده در گروه کنترلی تحت روش ساده‌تری فراهم می‌شوند که تحت این خاصیت صورت می‌گیرد.	<b> removeButton</b>

```
ButtonGroup {
    buttons: column.children
}

Column {
    id: column
    spacing: 5
    anchors.centerIn: parent

    Button {
        text: qsTr("C++")
    }
}
```

```

        text: qsTr("QML")
    }

Button {
    text: qsTr("HTML5")
}
}

```

به طور ذاتی این کنترل تمامی کنترل‌های پایه و Qt Quick را پشتیبانی می‌کند. همچنین، خاصیت `buttons` می‌تواند شناسه خاصی را جهت پذیرش و نمایش بپذیرد. با وجود کنترل `Column` و سه کنترل از نوع `Button` دسته بندی شده‌اند که نتیجه آن به صورت زیر خواهد بود:



## CheckBox کنترل

کنترل بعدی که از نامش مشخص است یک کنترل حالت انتخابی است که تحت کد زیر می‌توان از خاصیت آن استفاده کرد:

صفت	نوع
<code>checkState</code>	enumeration
<code>tristate</code>	bool

کنترل انتخاب در حالت عادی	<input type="checkbox"/> Normal
کنترل در حالت انتخاب شده	<input checked="" type="checkbox"/> Checked
کنترل در حالت متمرکز شده آماده انتخاب	<input type="checkbox"/> Focused
کنترل در حالت خاصیت غیرفعال	<input type="checkbox"/> Disabled

```

ColumnLayout {
    CheckBox {

```

```

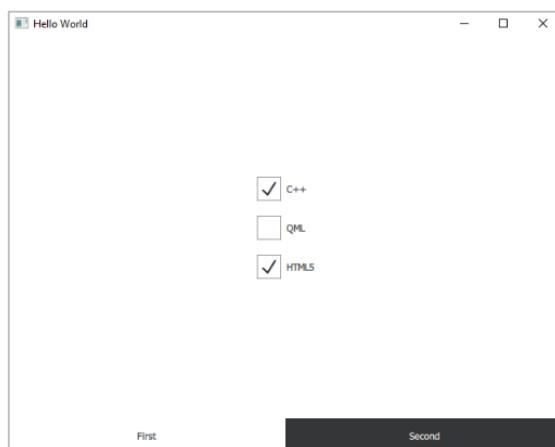
checked: true
text: qsTr("C++")

}

CheckBox {
    text: qsTr("QML")
}
CheckBox {
    checked: true
    text: qsTr("HTML5")
}
}

```

این کنترل می‌تواند از قبل با خاصیت `checked` با نوع بولین `true` و `false` مقدار دهی شود. در صورتی که خاصیت مربوطه تعیین نشود مقدار آن را به صورت خودکار از نوع `false` شناسایی خواهد کرد.



علاوه بر صفات‌های اصلی در رابطه با این کنترل می‌توان به ثبات‌هایی اشاره کرد که هم در دستورات شرطی می‌تواند مورد استفاده قرار گیرد و هم تحت صفت `checkState` پیکربندی کنترل را اعمال می‌کند.

توضیحات	ثابت‌ها
وضعیت کنترل در حالت انتخاب نشده.	<code>Qt.Unchecked</code>
وضعیت کنترل را در حالت تقریباً انتخاب شده مشخص می‌کند و البته زمانی مورد استفاده قرار می‌گیرد که صفت <code>tristate</code> فعال باشد.	<code>Qt.PartiallyChecked</code>
وضعیت کنترل را در حالت انتخاب شده قرار می‌دهد.	<code>Qt.Checked</code>

معرفی صفت `tristate` : این یک صفت ویژه جهت اعمال سه وضعیت‌های انتخاب شده، انتخاب نشده و وضعیت نا مشخص. در حالت عادی وضعیت اول و دوم موجود بوده و می‌توانیم آن را بر روی دکمه انتخابی اعمال کنیم.

اما در صورتی که نیاز باشد از وضعیت سوم استفاده شود نیاز است تا خاصیت tristate را فعال نماییم که در این صورت نیز ثابت طبق مثال زیر قابل استفاده خواهد بود:

```
CheckBox {
    text: qsTr("C++")
    tristate: true
    checkState : Qt.PartiallyChecked
}
```

### استفاده از کنترل CheckDelegate

این کنترل همانند کنترل قبلی بوده و با تفاوت اینکه می‌توان از آن در کنترل‌های دیگری استفاده و ادغام کرد. برای مثال فرض کنید لیستی از آیتم‌ها را در یک [ListView](#) قرار داده‌ایم برای اینکه بتوانیم قابلیت انتخابی برای آن‌ها در نظر بگیریم استفاده از این کنترل گزینه مناسبی خواهد بود.

صفت	نوع
checkState	enumeration
tristate	bool

```
ListView {
    anchors.centerIn: parent
    width: 200
    height: 200
    model: ["C++", "QML", "HTML5"]

    delegate: CheckDelegate {
        text: modelData
    }
}
```

همانطور که مشاهده می‌کنید کنترل [ListView](#) شامل یک خصیصه [model](#) است که توسط آن مقادیر لیست شده را دریافت و توسط خاصیت [delegate](#) نوع و نحوه نمایش آن را تعیین می‌کند. در اینجا استفاده از [CheckDelegate](#) سبب می‌شود داده‌های ارسال شده چه تحت QML و چه تحت C++ در قالب لیست انتخابی ظاهر شوند. جهت سفارشی سازی و درک بهتر کار با لیست در نوبه‌ی خود توضیحات تکمیلی در رابطه با این کنترل ارائه خواهد گردید.

علاوه بر صفت‌ها اصلی در رابطه با این کنترل می‌توان به ثبات‌هایی اشاره کرد که هم در دستورات شرطی می‌تواند مورد استفاده قرار گیرد و هم تحت تحت صفت [checkState](#) پیکربندی کنترل را اعمال می‌کند.

توضیحات	ثابت‌ها
وضعیت کنترل در حالت انتخاب نشده.	Qt.Unchecked
وضعیت کنترل را در حالت تقریباً انتخاب شده مشخص می‌کند و البته زمانی مورد استفاده قرار می‌گیرد که صفت <a href="#">tristate</a> فعال باشد.	Qt.PartiallyChecked
وضعیت کنترل را در حالت انتخاب شده قرار می‌دهد.	Qt.Checked

معرفی صفت **tristate** : این یک صفت ویژه جهت اعمال سه وضعیت در این کنترل است. برای مثال وضعیت‌های انتخاب شده، انتخاب نشده و وضعیت نامشخص. در حالت عادی وضعیت اول و دوم موجود بوده و می‌توانیم آن را بر روی دکمه انتخابی اعمال کنیم.

اما در صورتی که نیاز باشد از وضعیت سوم استفاده شود نیاز است تا خاصیت tristate را فعال نماییم که در این صورت نیز ثابت طبق مثال زیر قابل استفاده خواهد بود:

```
CHECKBOX {
    TEXT: QSTr("C++")
    TRISTATE: TRUE
    CHECKSTATE : Qt.PartiallyChecked
}
```

## استفاده از کنترل ComboBox

کنترل لیست انتخابی می‌تواند لیستی از آیتم‌ها را برای انتخاب در خود جای دهد.

نوع	صفت
int	<b>count</b>
int	<b>currentIndex</b>
string	<b>currentText</b>
Component	<b>delegate</b>
string	<b>displayText</b>
int	<b>highlightedIndex</b>
item	<b>indicator</b>
model	<b>model</b>
Popup	<b>popup</b>
bool	<b>pressed</b>
string	<b>textRole</b>
<b>توضیح عملکرد سیگنال</b>	
این سیگنال زمانی ساطع می‌شود که آیتم موجود در فهرست توسعه کاربر فعل شود.	<b>activated</b>
این خاصیت شماره شاخص (ایندکس) آیتم‌های برجسته شده را در لیست نگه می‌دارد.	<b>hilighted</b>
<b>توضیح عملکرد تابع</b>	
شاخص جعبه‌ایتم، و یا شاخص‌های برجسته شده زمانی که آن‌ها قابل مشاهده هستند را کاهش می‌دهد.	<b>decrementCurrentIndex</b>
شاخص (ایندکس) از متن مشخص شده را برگشت می‌دهد، یا در صورت برگشت دادن مقدار ۱- بیانگر این است که نتیجه‌ای یافت نشد.	<b>find</b>
شاخص جعبه‌ایتم، و یا شاخص‌های برجسته شده زمانی که آن‌ها قابل مشاهده هستند را افزایش می‌دهد.	<b>incrementCurrentIndex</b>
متن شاخص مشخص شده را بر می‌گرداند، یا یک متن خالی را در صورت خارج از محدوده بودن شاخص ارسال خواهد کرد.	<b>textAt</b>

برای استفاده از آن کافی است به صورت زیر کد مورد نظر را بنویسیم.

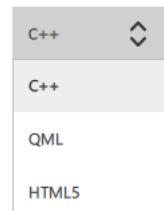
```
ComboBox {
```

```

    model: ["C++", "QML", "HTML5"]
}

```

این کنترل دارای خاصیت **model** است که علاوه بر پذیرش داده به صورت دستی می‌تواند از طریق C++ نیز داده‌ها را به راحتی قبول کرده و نمایش دهد. بعد از اجرای کد نتیجه به صورت زیر خواهد بود:



## استفاده از کنترل Container

کنترل فوق یک نوع پایه از کنترل‌های رابط‌کاربری است که اجازه حذف یا درج آیتم را به صورت پویا می‌دهد.

نوع	صفت
list<Item>	<b>contentChildren</b>
list<Object>	<b>contentData</b>
model	<b>contentModel</b>
int	<b>count</b>
int	<b>currentIndex</b>
Item	<b>currentItem</b>
توضیح عملکرد تابع	عملکرد
یک آیتم را اضافه می‌کند.	<b>addItem</b>
به شاخص جاری از نگه دارنده می‌افزاید.	<b>decrementCurrentIndex</b>
از شاخص جاری نگه دارنده می‌کاهد.	<b>incrementCurrentIndex</b>
یک آیتم را در فهرست اضافه می‌کند.	<b>insertItem</b>
آیتم موجود در فهرست را برابر می‌گرداند، اگر موردی یافت نشود مقدار خالی "null" را برابر می‌گرداند.	<b>itemAt</b>
یک آیتم را از یک شاخص (این‌دیکس) به یک شاخص (این‌دیکس) دیگر منتقل می‌کند.	<b>moveItem</b>
یک آیتم را از فهرست حذف می‌کند.	<b>removeItem</b>

```

Container {
    id: container
    anchors.fill: parent

    contentItem: ListView {
        model: container.contentModel
    }

    Text {
        text: "Text 1"
    }
}

```

```

width: container.width
height: container.height
}

Text {
    text: "Text 2"
    width: container.width
    height: container.height
}

Button {
    text: "Button 1"
    width: 220
    height: 35
}
}

```

این کنترل رابطهای برنامه‌نویسی برای افزودن، قرار دادن، حرکت و حذف کردن آیتم‌ها به صورت پویا دارا است، که توسط خاصیت itemAt یا contentChildren دسترسی به آیتم‌های این کنترل میسر می‌شود. بسیاری از ظروف (کانتینرها) دارای خاصیت‌های آیتم جاری (current) هستند. این خاصیت شاخص یا شناسهٔ جاری از صفت را مشخص می‌کند و می‌تواند به صورت فقط خواندنی آن را در اختیار صفت currentItem قرار دهد.

## کنترل Control

این کنترل نوع پایه و اصلی از کنترل رابطکاربری است، با دریافت رویدادهای ورودی از سیستم پنجره، نتیجهٔ آن را بر روی صفحه نمایش رسم می‌کند. در واقع تمامی کنترل‌ها در نهایت زیر مجموعه‌ای از این کنترل اصلی بشمار خواهند آمد. نوع مطرحی از این کنترل نمونه اولی است که مثل زده شد و با نام ApplicationWindow می‌توان از آن یاد کرد.

نوع	صفت
real	availableHeight
real	availableWidth
Item	background
real	buttonPadding
Item	contentItem
enumeration	focusPolicy
enumeration	focusReason
font	font
bool	hoverEnabled
bool	hovered
real	leftPadding
Locale	locale
bool	mirrored
real	padding
real	rightPadding
real	spacing
real	topPadding

bool	<b>visualFocus</b>
bool	<b>wheelEnabled</b>
<b>توضیح عملکرد تابع</b>	<b>عملکرد</b>
یک آیتم را اضافه می‌کند.	<b>addItem</b>
به شاخص جاری از نگه دارنده می‌افزاید.	<b>decrementCurrentIndex</b>
از شاخص جاری نگه دارنده می‌کاهد.	<b>incrementCurrentIndex</b>
یک آیتم را در فهرست اضافه می‌کند.	<b>insertItem</b>
آیتم موجود در فهرست را برابر می‌گرداند، اگر موردی یافت نشود مقدار خالی "null" را برابر می‌گرداند.	<b>itemAt</b>
یک آیتم را از یک شاخص (ایندکس) به یک شاخص (ایندکس) دیگر منتقل می‌کند.	<b>moveItem</b>
یک آیتم را از فهرست حذف می‌کند.	<b>removeItem</b>



## Dial کنترل

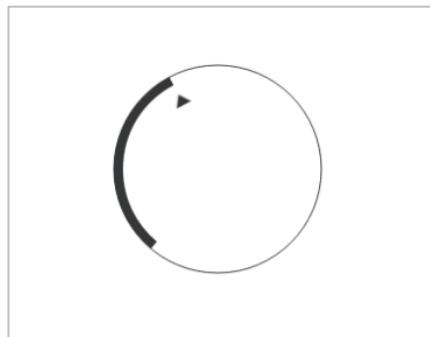
این کنترل شبیه به یک دستگاه شماره گیر سنتی است، در دستگاه‌های مانند تجهیزات استریو، دستگاه‌های صنعتی و غیره... می‌توان نمونه‌های مشابه آن را یافت. این کنترل امکان انتخاب بازه‌ای از مقادیر (ارزش) را در یک محدوده خاص برای کاربر فراهم می‌کند. برای مثال می‌توان مقدار مشخصی را با صفت `value` انتخاب کرد مثلاً مقدار ۱ تا ۱۰۰ را که توسط صفات‌های از (from) با مقدار ۰ و تا (to) با مقدار ۱۰۰ محدوده مشخص و مجاز را برای آن در نظر گرفت.

نوع	صفت
real	<b>angle</b>
real	<b>from</b>
component	<b>handle</b>
real	<b>position</b>
bool	<b>pressed</b>
enumeration	<b>snapMode</b>
real	<b>stepSize</b>
real	<b>to</b>
real	<b>value</b>
bool	<b>wrap</b>
<b>توضیح عملکرد تابع</b>	<b>عملکرد</b>
کاهش دهنده ارزش / مقدار توسط خاصیت <code>stepSize</code> و یا مقدار ۰.۱ در صورتی که خصیصه <code>stepSize</code> تعریف نشده باشد.	<b>decrease</b>
افزایش دهنده ارزش / مقدار توسط خاصیت <code>stepSize</code> و یا مقدار ۰.۱ در صورتی که خصیصه <code>stepSize</code> تعریف نشده باشد.	<b>increase</b>

کد زیر نحوه استفاده از این کنترل را نمایش می‌دهد که نتیجه آن طبق تصویر بعدی خواهد بود:

### Dial {

```
from: 0  
to: 100  
value: 14  
}
```



## کنترل Drawer

این کنترل که با اصطلاح کشو، مبتنی بر موقعیت‌های سمت راست، چپ و حتی بالا و پایین معروف است. بیشتر، در سیستم‌های لمسی به کار می‌رود و یک محل مناسبی را برای منوها یا ابزارهای کاربردی ارائه می‌کند. روش استفاده از آن با فراخوانی نام کنترل و تخصیص مقدار به صفات height و width بر اساس شناسه است.

نوع	صفت
real	dragMargin
enumeration	edge
real	position

### Drawer {

```
id: drawer  
width: 0.66 * window.width  
height: window.height  
}
```

البته به خاطر خاصیتی که این کنترل دارد برای اینکه به درستی و با دقیق‌تر مشاهده شود می‌بایست مقدار width آن را کمی بیشتر در نظر بگیریم تا فضای مورد نظر جهت تقسیم‌بندی ایجاد شود.

```

import QtQuick 2.9
import QtQuick.Controls 2.2

ApplicationWindow {
    id:window
    width: 200
    height: 228
    visible: true
    title: qsTr("Hello World")

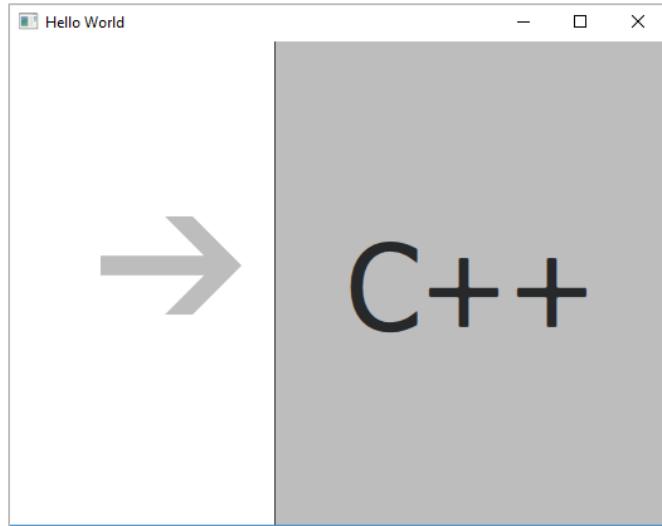
    Drawer {
        id: drawer
        width: 0.66 * window.width
        height: window.height
    }

    Label {
        id: content
        text: "C++"
        font.pixelSize: 96
        anchors.fill: parent
        verticalAlignment: Label.AlignVCenter
        horizontalAlignment: Label.AlignHCenter
    }
}

```

این کنترل یک ویژگی پر کاربردی را دارد که اجازه می‌دهد تا حالت کشوئی آن در چهار لبهٔ مورد نظر سفارشی شود که در زیر مشخص شده است:

توضیحات	ثابت ها
نمایش در لبه بالایی از محتوای صفحه	Qt.TopEdge
نمایش در لبه سمت چپ از محتوای صفحه گزینه فعال در حالت پیش فرض)	Qt.LeftEdge
نمایش در لبه سمت راست از محتوای صفحه	Qt.RightEdge
نمایش در لبه پایینی از محتوای صفحه	Qt.BottomEdge



## کنترل Frame

این کنترل به عنوان یک قاب برای کنترل‌گرهای منطقی مورد استفاده قرار می‌گیرد. اما برای مدیریت زیر آیتم‌های آن، ممکن است از دو آیتم دیگر `ColumnLayout` و یا یک `RowLayout` استفاده شود.

البته هرچند این یک کنترل از مازول Qt Quick است اما توجه داشته باشید که به خاطر نیاز به دو لایه ذکر شده باید مازول لایه (Layouts) را به روش زیر وارد سند QML نمایید:

```
import QtQuick.Layouts 1.3

Frame {
    ColumnLayout {
        anchors.fill: parent
        CheckBox { text: qsTr("C++") }
        CheckBox { text: qsTr("QML") }
        CheckBox { text: qsTr("JavaScript") }
    }
}
```

کد و نتیجه نهایی از این کنترل به صورت زیر خواهد بود :

```
import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.2

ApplicationWindow {
    id:window
    width: 200
    height: 228
    visible: true
    title: qsTr("Hello World")
```

```

Frame {
    ColumnLayout {
        anchors.fill: parent
        CheckBox { text: qsTr("C++") }
        CheckBox { text: qsTr("QML") }
        CheckBox { text: qsTr("JavaScript") }
    }
}

```



## کنترل **GroupBox**

این کنترل به عنوان یک قاب همانند کنترل قبلی Frame برای کنترل گروهی از کنترل‌های منطقی مورد استفاده قرار می‌گیرد. اما برای مدیریت زیر آیتم‌های آن باید از دو آیتم دیگر **RowLayout** و یا یک **ColumnLayout** استفاده شود.

البته هرچند این یک کنترل از مازول Qt Quick است اما توجه داشته باشید که به خاطر نیاز به دو لایه ذکر شده باید مازول لایه (Layouts) را به روش زیر وارد سند QML نمایید:

صفت	نوع
<b>label</b>	Item
<b>title</b>	string

```
import QtQuick.Layouts 1.3
```

```

GroupBox {
    title: qsTr("Select Programming Language")
    ColumnLayout {
        anchors.fill: parent

        CheckBox { text: qsTr("C++") }
        CheckBox { text: qsTr("QML") }
        CheckBox { text: qsTr("JavaScript") }
    }
}

```



نتیجهٔ فوق با تفاوت وجود یک عنوان نسبت به **Frame** به شرح مقابل است:

البته توجه داشته باشید که خاصیت‌هایی برای سفارشی سازی موجود است که می‌توان کنترل را به صورت ویژه توسعه داد که در ادامه کد و تصویر نهایی مشخص شده است:

```

import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.2

ApplicationWindow {
    id:window
    width: 200
    height: 228
    visible: true
    title: qsTr("Hello World")

    GroupBox {
        anchors.centerIn: parent
        label: CheckBox {
            id: checkbox

            checked: true
            text: qsTr("Select Programming Language")
        }

        ColumnLayout {
            anchors.fill: parent
            enabled: checkBox.checked
            CheckBox { text: qsTr("C++") }
            CheckBox { text: qsTr("QML") }
            CheckBox { text: qsTr("JavaScript") }
        }
    }
}

```

با توجه به کد فوق، متوجه خواهید شد که کنترل `GroupBox` با استفاده از صفت سفارشی ساز با نام `label` می‌تواند نوع شیء فرزند را فراخوانی کند که قبلاً راجع به این صفت توضیحات داده شده است. در این بخش از کد یک کنترل انتخابی ایجاد شده و با بررسی وضعیت آن توسط خود کنترل انتخاب با شناسه `checkbox` وضعیت فعال یا غیرفعال بودن لایه `ColumnLayout` مشخص می‌شود. در واقع این روشی برای اعمال و ترکیب است که دسترسی به اشیاء زیر والد توسط لایه مذکور تغییر پیدا می‌کند و نتیجه نهایی به صورت زیر خواهد بود:



## كنترل ItemDelegate

این کنترل آیتم‌هایی را به عنوان لیست نمایشی فراهم می‌کند. این قابلیت می‌تواند در کنترل‌های مختلفی همچون `ComboBox` و `ListView` مورد استفاده قرار بگیرد که دارای یک صفت ویژه با نام `highlighted` است که می‌توان با سفارشی سازی آن وضعیت آیتم انتخاب شده را تغییر داد.

صفت	نوع
<code>highlighted</code>	<code>bool</code>

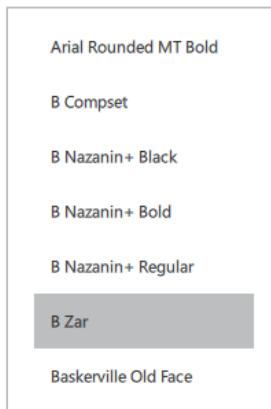
روش استفاده از آن به صورت زیر است :

```
delegate: ItemDelegate {  
    text: modelData  
    width: parent.width  
}
```

به عنوان مثال در نظر بگیرید می‌خواهیم لیستی از آیتم‌ها را توسط این کنترل دریافت و در داخل یک کنترل با قابلیت نمایش لیستی از آیتم‌ها ارسال کنیم. جهت مشاهده این قابلیت، لیست قلم‌ها (فونت‌های) نصب شده بر روی سیستم را جمع‌آوری و به نمایش در آوریم که با انتخاب هر یک از آن‌ها نام آن در کنسول محیط توسعه چاپ شود.

```
ListView {  
    width: 160  
    height: 240  
    model: Qt.fontFamilies()  
    delegate: ItemDelegate {  
        text: modelData  
        width: parent.width  
    }  
    ScrollIndicator.vertical: ScrollIndicator { }  
}
```

کد مربوطه به عنوان یک شیء فرزند به شیء والد که یک کنترل `ListView` است اشاره دارد که صفت متن یا همان (`text`) با دریافت مدل داده از لیست فونت‌ها آن را نمایش خواهد داد. تصویر نهایی آن به صورت زیر خواهد بود :



همچنین کد نهایی را در ادامه مشاهده می‌کنید:

```

import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.2

ApplicationWindow {
    id:window
    width: 200
    height: 228
    visible: true
    title: qsTr("Hello World")

    ListView {
        width: 160
        height: 240

        model: Qt.fontFamilies()

        delegate: ItemDelegate {
            text: modelData
            width: parent.width
            onClicked: console.log("Your selected font name is : ", modelData)
        }
    }

    ScrollIndicator.vertical: ScrollIndicator { }

}

}

```

با انتخاب آیتم مورد نظر نام (متن - عنوان) آیتم انتخاب شده که در اینجا فونت (قلم) است چاپ خواهد شد:

qml: Your selected font name is : B Zar

## کنترل Label

کنترل برچسب، متنی با ظاهر طراحی شده و ارث بری از فونت و با داشتن رنگ و ویژگی پیش فرض و خاصی یکی از مناسب‌ترین کنترل‌ها به عنوان استفاده از برچسب بر روی فرم است.

صفت	نوع
background	Item

فرم کلی آن به صورت زیر است:

```
Label {
    text: "برچسب"
}
```

برای سفارشی سازی آن می‌توان از خصیصه‌های متن (text)، فونت (font) و غیره استفاده کرد که در زیر آمده است:

```
Label {
    text: "My name is Kambiz"
    font.pixelSize: 16
    font.italic: true
    color: "#f23"
}
```

## کنترل Menu

این کنترل به دو دلیل مورد استفاده قرار می‌گیرد، یک (منوی زمینه شده) منوی که با راست کلیک شدن ظاهر می‌شود. و دوم منوی پنجره‌ای، منوی که با کلیک شدن بر روی یک دکمه ظاهر می‌شود. توجه نمایید که آیتم‌های ذکر شده توسط کنترل بعدی یعنی MenuItem استفاده می‌شود.

صفت	نوع
contentData	list<Object>
contentModel	model
title	string
سیگنال	توضیح عملکرد سیگنال
addItem	آیتمی را به آخر لیست آیتم‌های موجود اضافه می‌کند.
insertItem	آیتمی را در فهرست قرار می‌دهد.
itemAt	آیتمی را از فهرست بر می‌گرداند، در صورتی که موردی یافت نشود مقدار خالی "null" را بر می‌گرداند.
moveItem	یک آیتم را از یک شاخص (ایندهکس) به یک شاخص (ایندهکس) دیگر منتقل می‌کند.
removeItem	آیتمی را از فهرست حذف می‌کند.

```
Button {
    id: fileButton
    anchors.centerIn: parent
    text: "File"
    onClicked: menu.open()

    Menu {
        id: menu
        y: fileButton.height

        MenuItem {
```

```

        text: "New..."
    }

MenuItem {
    text: "Open..."
}
MenuItem {
    text: "Save"
}
}

}

```

در کد ذکر شده با استفاده از خاصیت رویداد کلیک شدن کنترل **Button** منوی تعریف شده توسط کنترل **Menu** با ۳ آیتم نمایان می‌شود. همچنین کد مرتبط به مختصات y مقدار ارتفاع دکمه را گرفته و به اندازه، همان منو را در محدوده محور y فاصله می‌دهد تا هنگام نمایش دکمه و منو از هم جدا دیده شوند.

## MenuItem کنترل

این کنترل همانند **Button** رخدادهایی را دارد که در اختیار کنترل **Menu** قرار می‌دهد. توسط این کنترل می‌توان مقادیر قابل عملیاتی را به عنوان آیتم برای منو ایجاد کرد.

```

Menu {
    id: menu
    y: fileButton.height

MenuItem {
    text: "New..."
    onTriggered: ...
}

MenuItem {
    text: "Open..."
    onTriggered: ...
}
MenuItem {
    text: "Save"
    onTriggered: ...
}
}

```

## Page کنترل

این کنترل با خاصیت ظرف (Container) خود به راحتی یک محدوده‌ای را فراهم می‌کند که پا ورقی و سربرگ صفحه را مدیریت کند.



به عنوان مثال در ادامه تکه کد ذکر شده نشان می‌دهد که چگونه با استفاده از یک نوار ابزار مشخص می‌توان بخش گسترهای از بالا و پایین صفحه را مدیریت کرد.

صفت	نوع
<b>contentChildren</b>	list<Item>
<b>contentData</b>	list<Object>
<b>header</b>	real
<b>footer</b>	real
<b>title</b>	real

توجه داشته باشید که بهترین راهکار برای استفاده بهتر از این کنترل ترکیب آن با کنترل دیگری مانند StackView خواهد بود که در صفحات بعد به آن اشاره خواهیم کرد.

```
StackView {
    anchors.fill: parent

    initialItem: Page {
        header: ToolBar {
            // ...
        }

        contentData: Item {
        }

        contentChildren : Item {
        }
    }

    footer:
        TabBar {
            // ...
        }
    }
}
```

## PageIndicator کنترل

این کنترل زمانی مورد استفاده می‌گیرد که نیاز باشد طبق یک جلوه بصری صفحه جاری را روی فرم مشخص کنید. فرض نحوه استفاده از آن به صورت زیر است:

صفت	نوع
<b>count</b>	int
<b>currentIndex</b>	int
<b>delegate</b>	Component

interactive	bool
-------------	------

```
PageIndicator {
    count: 3
    currentIndex: 2
}
```

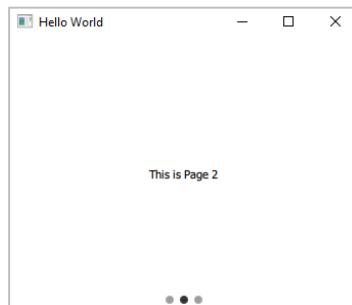
جهت استفاده از این کنترل می‌بایست آن را توسط کنترل دیگری با نام SwipeView ترکیب کنید تا بتوانید از صفت‌های آن استفاده و نشانگر صفحه جاری توسط این کنترل را فعال نمایید. در کد زیر به نحوه استفاده از این کنترل اشاره می‌شود:

```
SwipeView {
    id: view
    currentIndex: 1
    anchors.fill: parent
    Item {
        id: firstPage
    }
    Item {
        id: secondPage
    }
    Item {
        id: thirdPage
    }
}
PageIndicator {
    id: indicator
    count: view.count
    currentIndex: view.currentIndex
    interactive: true
    anchors.bottom: view.bottom
    anchors.horizontalCenter: parent.horizontalCenter
}
```

در صورتی که به بدنۀ PageIndicator توجه کنید متوجه خواهید شد که صفت‌های count به تعداد صفحات موجود توسط کنترل SwipeView مذکور در کنترل currentIndex برابر خاصیت currentIndex است این در اشاره دارد و تعداد صفحات آن را دریافت می‌کند. سپس خاصیت currentIndex کنترل SwipeView است این در حالی است که مقدار خاصیت مذکور در کنترل والد برابر ۱ است در واقع بعد از اجرا خواهیم دید که کنترل PageIndicator با فعال‌سازی گزینه دوم نشان دهنده صفحه دوم خواهد بود.

توجه داشته باشید که شمارنده در این کنترل طبق قوانین اصلی از آغاز می‌شود.

همچنین خاصیت **interactive** در صورتی که با یک مقدار `true` برابر باشد زمان کلیک بر روی نشانگر تعداد صفحات و اکتشن نشان خواهد داد.



## Pane کنترل

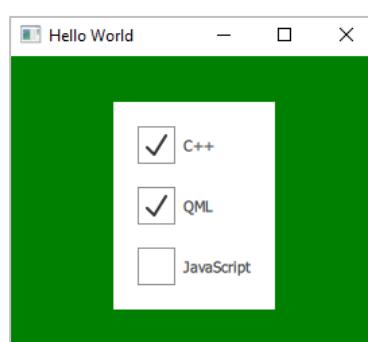
این کنترل رنگ پس‌زمینه‌ای را برای ادغام و یکسان شدن با سبک برنامه فراهم می‌کند. این کنترل از خود هیچ لایه‌ای ایجاد نمی‌کند، اما برای مدیریت هرچه بهتر لازم است با کنترل‌های `ColumnLayout` و یا یک `RowLayout` ترکیب شود. شکل کلی آن به صورت زیر است:

صفت	نوع
<code>contentChildren</code>	<code>list&lt;Item&gt;</code>
<code>contentData</code>	<code>list&lt;Object&gt;</code>
<code>contentHeight</code>	<code>real</code>
<code>contentWidth</code>	<code>real</code>

### Pane {

```
ColumnLayout {  
    anchors.fill: parent  
  
    CheckBox { text: qsTr("C++") }  
  
    CheckBox { text: qsTr("QML") }  
  
    CheckBox { text: qsTr("JavaScript") }  
}
```

کد فوق با ترکیب کنترل لایه‌ای می‌تواند خروجی زیر را فراهم کند:



## Popup کنترل

این کنترل یک نوع پایه از کنترل‌های رابط کاربر است که می‌توان آن را با کنترل پنجره (Window) یا ApplicationWindow استفاده کرد. خاصیت پنجره کوچک (Popup) ویزگی اصلی آن است که با افکت دارای سایه و شفافیت پشت سر خودش بر روی پنجره اصلی ظاهر می‌شود.

نوع	صفت
bool	activeFocus
real	availableHeight
real	availableWidth
Item	background
real	buttonMargin
real	buttonPadding
bool	clip
enumeration	closePolicy
enumeration	contentChildren
enumeration	contentData
real	contentHeight
item	contentItem
real	contentWidth
font	font
bool	dim
Transition	enter
Transition	exit
bool	focus
real	height
real	implicitHeight
real	implicitWidth
bool	hovered
real	leftMargin
real	leftPadding
Locale	locale
bool	mirrored
real	margins
real	padding
real	rightMargin
real	rightPadding
bool	modal
real	scale
real	spacing
item	parent
real	opacity
real	topPadding
real	topMargin
enumeration	transformOrigin
bool	visible
real	width
real	x
real	y
real	z
توضیح عملکرد سیگنال	
زمانی که پنجره بسته می‌شود این سیگنال ساطع می‌شود.	
	سیگنال
	closed

زمانی که پنجره باز می‌شود این سیگنال ساطع می‌شود.	<b>opened</b>
<b>توضیح عملکرد تابع</b>	عملکرد
بستن پنجره	<b>close</b>
پنجره را به عنوان نمایش یک دلیل فعال نگه می‌دارد.	<b>forceActiveFocus</b>
باز کردن پنجره	<b>open</b>

ساختار کلی این کنترل به صورت زیر است:

```
Popup {
    id: popup

    x: 100
    y: 100
    width: 200
    height: 300
    modal: true
    focus: true
    closePolicy: Popup.CloseOnEscape | Popup.CloseOnPressOutsideParent
}
```

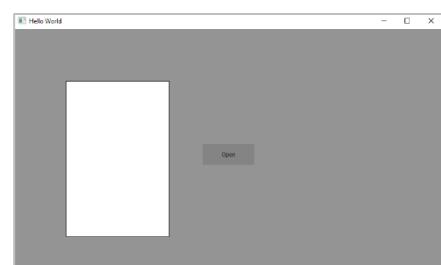
کد زیر با کنترل ApplicationWindow ترکیب شده است :

```
import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.2

ApplicationWindow {
    id: window
    width: 200
    height: 228
    visible: true
    title: qsTr("Hello World")

    Button {
        text: "Open"
        onClicked: popup.open()
    }
}
```

```
Popup {
    id: popup
    x: 100
    y: 100
    width: 200
    height: 300
```



```

    modal: true
    focus: true
    closePolicy: Popup.CloseOnEscape | Popup.CloseOnPressOutsideParent
}
}

```

## ProgressBar کنترل

این کنترل به عنوان یک نوار پیشرفت از وضعیت عملیات را فراهم می‌کند. مقدار آن به طور منظم باید به روز رسانی شود، محدوده آن می‌تواند توسط کاربر سفارشی سازی و توسط صفات‌های از (from) و تا (to) مشخص شود که هر یک می‌توانند مقادیری را دریافت کنند.

صفت	نوع
from	real
indeterminate	bool
position	real
to	real
value	real
visualPosition	real

ساختار کلی و ظاهر کنترل در حالت عادی و فیر فعال به صورت زیر است:

```

ProgressBar {
    value: 0.5
}

```



یک ویژگی خاص در این کنترل موجود است که اجازه می‌دهد حالت نامشخصی از وضعیت را برای بیننده فراهم کند. برای مثال زمانی که مشخص نیست، وضعیت نهایی و باقی مانده از یک فرآیند در چه مرحله‌ای است می‌توان از این قابلیت استفاده کرد که با صفت indeterminate مشخص می‌شود و ظاهر کلی آن در این وضعیت به صورت زیر است:



کد زیر با کنترل‌های افزایش دهنده و کاهش دهنده ترکیب شده است:

```

ColumnLayout {
    anchors.centerIn: parent
    Button {
        id:plus
        text: "+"
        onClicked: myProgressBar.value = myProgressBar.value + 1
    }
}

```

```

}

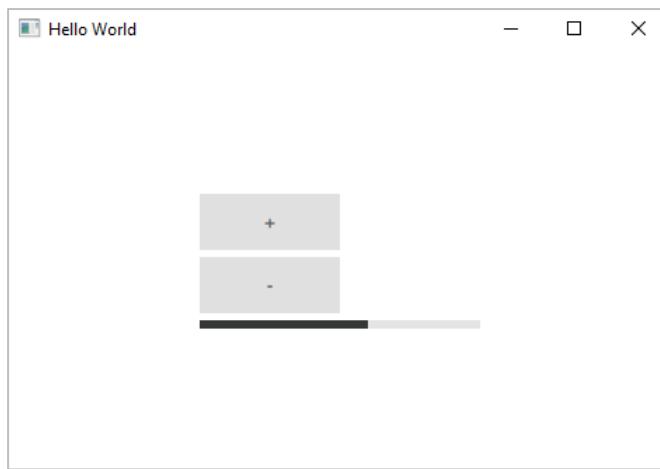
Button {
    id:mines
    text: "-"
    onClicked: myProgressBar.value = myProgressBar.value - 1
}

ProgressBar {
    id:myProgressBar
    to: 10
    from: 0
    value: 1
}

}

```

در نهایت ظاهر مربوطه به صورت زیر خواهد بود که با افزایش و کاهش مقدار نوار پیشرفت مواجه خواهیم شد:



## RadioButton کنترل

این کنترل امکان انتخاب گزینه‌ای را فراهم می‌سازد، می‌تواند با جایه‌جا شدن (انتخاب شدن یا نشدن) وضعیت روشن یا خاموش را مشخص کند. این کنترل معمولاً زمانی مورد استفاده قرار می‌گیرد که لازم است از بین چند گزینه قابل انتخاب یکی را انتخاب کنید.

وضعیت کنترل در حالت عادی	<input type="radio"/> Normal
وضعیت کنترل در حالت انتخاب شده	<input checked="" type="radio"/> Checked
وضعیت کنترل در حالت انتخاب نشده اما تمرکز شده	<input type="radio"/> Focused
وضعیت کنترل در حالت غیرفعال	<input type="radio"/> Disabled

این دکمه به صورت خودکار عمل می‌کند و تنها در بین چندین شیء فرزند تنها یکی از آن‌ها قابل انتخاب خواهد بود.

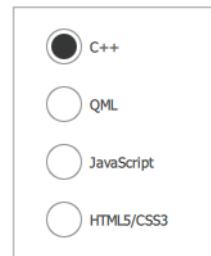
صفت	نوع
AbstractButton	مشتق شده از صفت‌های

ساختار کلی این کنترل به صورت زیر است:

```
RadioButton {  
    text: qsTr("C++")  
}
```

کد زیر با کنترل لایه‌ای به عنوان کنترل والد برای دسته بندی ترکیب شده است :

```
ColumnLayout {  
  
    RadioButton {  
        checked: true  
        text: qsTr("C++")  
    }  
    RadioButton {  
        text: qsTr("QML")  
    }  
    RadioButton {  
        text: qsTr("JavaScript")  
    }  
    RadioButton {  
        text: qsTr("HTML5/CSS3")  
    }  
}
```



## RadioDelegate کنترل

این کنترل همانند کنترل قبلی است و به عنوان یک نماینده برای استفاده در کنترل‌های دیگر است، و مانند `ListView` بکار گرفته می‌شود. تمامی ویژگی‌های آن مانند کنترل اصلی است و کد نمونه آن به صورت زیر آورده شده است:

```
ButtonGroup {  
    id: buttonGroup  
}  
ListView {  
    model: ["C++", "QML", "JavaScript"]  
    delegate: RadioDelegate {  
        text: modelData  
        checked: index == 0  
        ButtonGroup.group: buttonGroup  
    }  
}
```

## RangeSlider کنترل

همانطور که از نامش مشخص است برای انتخاب یک محدوده‌ای بین دو ارزش تعریف شده مورد استفاده قرار می‌گیرد که توسط یک نوار لغزنده میسر می‌شود.

وضعیت کنترل در حالت عادی	
وضعیت کنترل در حالتی که دکمه لغزنده اول مورد استفاده قرار می‌گیرد	
وضعیت کنترل در حالتی که دکمه لغزنده دوم مورد استفاده قرار می‌گیرد	
وضعیت دکمه در حالت غیرفعال	

قالب اصلی این کنترل به صورت زیر است:

```
RangeSlider {
    first.value: 0.25
    second.value: 0.75
}
```

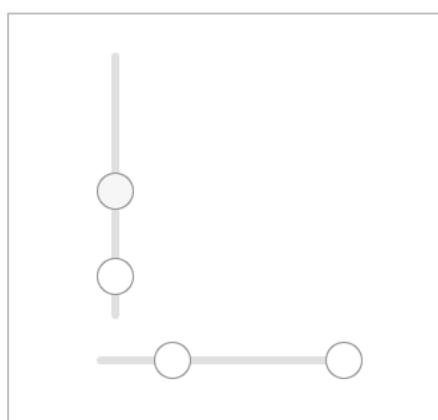
نوع	صفت
parent	<b>first</b>
real	<b>first.value</b>
real	<b>first.position</b>
real	<b>first.visualPosition</b>
item	<b>first.handle</b>
real	<b>first.pressed</b>
bool	<b>from</b>
enumeration	<b>orientation</b>
parent	<b>second</b>
real	<b>second.value</b>
real	<b>second.position</b>
item	<b>second.handle</b>
real	<b>second.pressed</b>
real	<b>first.visualPosition</b>
enumeration	<b>snapMode</b>
real	<b>stepSize</b>
real	<b>to</b>
توضیح عملکرد سیگنال	سیگنال
مقدار / ارزش دسته اول را توسط <b>stepSize</b> می‌کاهد. یا مقدار <b>۱</b> را در صورت عدم تعریف شدن <b>stepSize</b> اعمال می‌کند.	<b>first.decrease</b>
مقدار / ارزش دسته اول را توسط <b>stepSize</b> می‌افزاید. یا مقدار <b>۱</b> را در صورت عدم تعریف شدن <b>stepSize</b> اعمال می‌کند.	<b>first.increase</b>

مقدار / ارزش دسته دوم را توسط <code>stepSize</code> می کاهد، یا مقدار <code>1.0</code> را در صورت عدم تعریف شدن <code>stepSize</code> اعمال می کند.	<code>second.decrease</code>
مقدار / ارزش دسته دوم را توسط <code>stepSize</code> می افزاید، یا مقدار <code>1.0</code> را در صورت عدم تعریف شدن <code>stepSize</code> اعمال می کند.	<code>second.increase</code>
مقادیر دسته اول و دوم را فراهم می کند.	<code>setValues</code>

کد نهایی و خروجی آن به صورت زیر خواهد بود:

```
ColumnLayout {
    anchors.centerIn: parent
    RangeSlider {
        first.value: 0.25
        second.value: 0.75
        orientation: Qt.Vertical
    }
    RangeSlider {
        first.value: 0.25
        second.value: 0.75
        orientation: Qt.Horizontal
    }
}
```

در کد فوق کنترل مورد نظر با ترکیب لایه ای از `ColumnLayout` و تخصیص مقادیر بر خاصیت های `first.value` و `second.value` یک بازه ای از مقادیر را پذیرفته و آن را ارائه می کند. اما خاصیتی هم موجود است با نام `orientation` که می تواند با دریافت مقدار از پیش تعریف شده در `Qt` در محور عمودی و افقی وضعیت خود را ثابت نگه دارد.



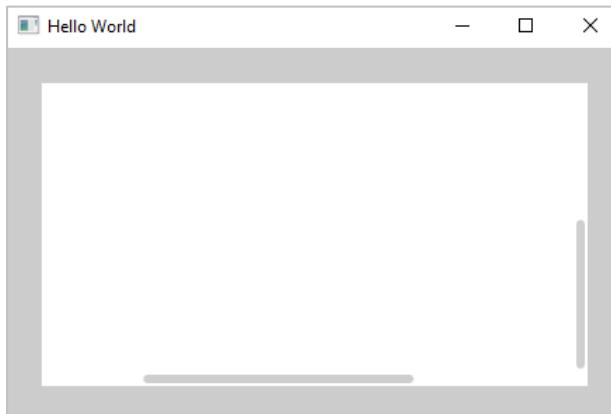
## كنترل ScrollBar

نوار اسکرول بار (نوار پیمایشی) یک کنترل تعاملی است که می تواند در صورت نیاز برای حرکت به یک موقعیت خاص، لیستی از آیتم ها یا حتی کنترل ها مورد استفاده قرار گیرد. این نوار به صورت عمودی و یا افقی و با پشتیبانی از ضمیمه شدن کنترل های دیگری مانند `ListView`, `Flickable`

و `GridView` کاربرد بسیار مهمی دارد. البته به این نکته توجه کنید که این کنترل به تنها یی قابل استفاده نیست و باید آن را به عنوان مکمل هدایت‌گر در کنترل‌های ذکر شده مورد استفاده قرار داد.

شکل کلی آن به صورت زیر است:

```
Flickable {
    // ...
    ScrollBar.vertical: ScrollBar { }
}
```



نوع	صفت
bool	<code>active</code>
enumeration	<code>orientation</code>
real	<code>position</code>
bool	<code>pressed</code>
real	<code>size</code>
real	<code>stepSize</code>
توضیح عملکرد سیگنال	سیگنال
این ویژگی به یک اسکرول بار افقی متصل می‌شود.	<code>horizontal</code>
این ویژگی به یک اسکرول بار عمودی متصل می‌شود.	<code>vertical</code>
توضیح عملکرد تابع	عملکرد
موقعیت را توسط <code>stepSize</code> کاهش می‌دهد، یا از ۰٪ به ۱٪ در صورتی که <code>stepSize</code> تعریف نشده باشد تغییر می‌دهد.	<code>decrease</code>
موقعیت را توسط <code>stepSize</code> افزایش می‌دهد، یا از ۰٪ به ۱٪ در صورتی که <code>stepSize</code> تعریف نشده باشد تغییر می‌دهد.	<code>increase</code>

کد نهایی و خروجی آن به صورت زیر خواهد بود:

```
Flickable {
```

```

anchors.fill: parent
contentWidth: parent.width * 2
contentHeight: parent.height * 2

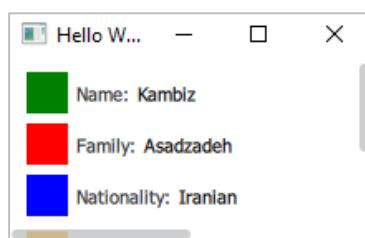
ScrollBar.horizontal: ScrollBar { id: hbar; active: vbar.active }
ScrollBar.vertical: ScrollBar { id: vbar; active: hbar.active }

}

```

در کد با ترکیب کنترل والد با نام `Flickable` و بکارگیری `ScrollBar` به عنوان یک شیء فرزند دسترسی به قابلیت همزمان اسکرول و لغزش فراهم شده است. البته این امکان به تنظیم خصیصه‌های `contentWidth` و `contentHeight` و `anchors.fill` هم بستگی دارد که به کاربرد و وظایف این خصیصه‌ها در صفحات آتی اشاره خواهد شد.

همچنین توجه داشته باشید که برای فعال‌سازی خصیصه‌های اسکرول در حالت عمودی و افقی می‌بایست از خصیصه‌های `vertical` و `horizontal` استفاده شود.



توجه داشته باشید که در این حالت حتی با گرفتن ماوس و حرکت دادن اشاره‌گر به جهت‌های مختلف مختلف بر روی صفحه هدایت می‌شود و همچنین بدون در نظر گرفتن محدودیت همیشه این امکان فعال خواهد بود. فرض کنیم نیاز است این امکان به صورت خودکار بر اساس محتوا تغییر کند و تنها اسکرول برای محور x قابل اجرا باشد.

برای اینکار کافی است صفت `contentWidth` را غیرفعال کنید و یا در کل آن را مورد استفاده قرار ندهید.

```

Flickable {

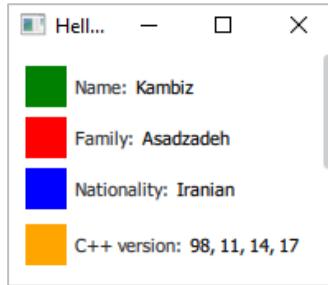
    anchors.fill: parent
    //contentWidth: parent.width * 2
    contentHeight: parent.height * 2

    ScrollBar.horizontal: ScrollBar { id: hbar; active: vbar.active }
    ScrollBar.vertical: ScrollBar { id: vbar; active: hbar.active }

}

```

در این صورت نتیجه به صورت دلخواه خواهد شد:



## ScrollIndicator کنترل

طبق روال این کنترل نیز به عنوان یک نمایه و مشتق شده از کنترل‌های قابل استفاده در کنترل‌های دیگری همچون `Flickable` و `ListView` است.

نوع	صفت
<code>bool</code>	<code>active</code>
<code>enumeration</code>	<code>orientation</code>
<code>real</code>	<code>position</code>
<code>bool</code>	<code>pressed</code>
<code>real</code>	<code>size</code>
<code>real</code>	<code>stepSize</code>
توضیح عملکرد سیگنال	سیگنال
این ویژگی به یک اسکرول بار افقی متصل می‌شود.	<code>horizontal</code>
این ویژگی به یک اسکرول بار عمودی متصل می‌شود.	<code>vertical</code>
توضیح عملکرد تابع	عملکرد
موقعیت را توسط <code>stepSize</code> کاهش می‌دهد. یا از <code>0, 0, 1, 0</code> در صورتی که <code>stepSize</code> تعریف نشده باشد تغییر می‌دهد.	<code>decrease</code>
موقعیت را توسط <code>stepSize</code> افزایش می‌دهد. یا از <code>0, 0, 1, 0</code> در صورتی که <code>stepSize</code> تعریف نشده باشد تغییر می‌دهد.	<code>increase</code>

## Slider کنترل

این کنترل نیز با داشتن یک دکمه به صورت لغزان امکان حرکت به صورت نواری لغزان در امتداد یک خط را فراهم می‌کند.  
شکل کلی این کنترل به صورت زیر است :

```
Slider {
    value: 0.5
}
```

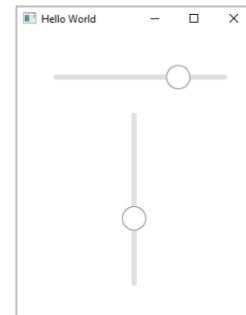
نوع	صفت
<code>real</code>	<code>from</code>
<code>enumeration</code>	<code>orientation</code>
<code>item</code>	<code>handle</code>
<code>real</code>	<code>position</code>

bool	<b>pressed</b>
enumeration	<b>snapMode</b>
real	<b>stepSize</b>
real	<b>to</b>
real	<b>value</b>
real	<b>visualPosition</b>
توضیح عملکرد سیگنال	سیگنال
کاهش دهنده ارزش / مقدار توسط خاصیت <b>stepSize</b> تعریف نشده باشد.	<b>decrease</b>
افزایش دهنده ارزش / مقدار توسط خاصیت <b>stepSize</b> و یا مقدار ۰.۱ در صورتی که خصیصه نشده باشد.	<b>increase</b>
مقدار موقعیت را برابر می‌گرداند.	<b>valueAt</b>



کد زیر نمونه مثالی است که می‌تواند این کنترل را مورد استفاده قرار دهد.

```
Slider {
    x: 37
    y: 32
    orientation: Qt.Horizontal
    to: 10
    value: 0
}
```



```
Slider {
    x: 116
    y: 87
    value: 0
    to: 10
    orientation: Qt.Vertical
}
```

}

در کد مذکور با تخصیص حداقل و حداکثر مقدار قابل ارزیابی، کنترل مورد نظر جهت استفاده ایجاد شده است.

اما جهت دریافت مقدار مربوطه در هنگام تغییر نوار لغزنده کافی است از رویداد **onValueChanged** در این کنترل استفاده شود.

**Text** {

```
    id: vText
    text: "0"
}
```

```

Slider {
    x: 116
    y: 87
    value: 0
    to: 10
    orientation: Qt.Vertical

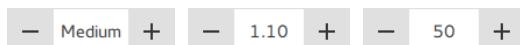
    onValueChanged: {
        vText.text = value
    }
}

```

تحت کد جاوااسکریپت مقدار انتخاب شده در متن ورودی vText که شناسه کنترل Text است منتقل خواهد شد.

## SpinBox کنترل

این کنترل اجازه این را می‌دهد تا کاربر در محدوده یک مقدار از پیش تعیین شده، مقدار مورد نظر خود را انتخاب کند. همچنین با فشرده شدن کلیدهای جهت به پایین و بالا بر روی صفحه کلید امکان انتخاب مقدار وجود دارد. این کنترل، ویرگی ویرایش شدن را دارد و کاربر می‌تواند در صورت عدم وجود مقدار مورد نظر در بازه از پیش تعریف شده آن را خود وارد کند. البته شکل کلی این کنترل به صورت زیر است:



```

SpinBox {
    value: 50
}

```

نوع	صفت
parent	down
bool	down.pressed
item	down.indicator
bool	editable
int	from
int	stepSize
function	textFromValue
int	to
parent	up
bool	up.pressed
item	up.indicator
Validator	validator

int	value
function	valueFromText
توضیح عملکرد سیگنال	سیگنال
کاهش دهنده ارزش / مقدار توسط خاصیت stepSize و یا مقدار ۰.۱ در صورتی که خصیصه stepSize تعریف نشده باشد.	<b>decrease</b>
افزایش دهنده ارزش / مقدار توسط خاصیت stepSize و یا مقدار ۰.۱ در صورتی که خصیصه stepSize تعریف نشده باشد.	<b>increase</b>

برخلاف ساده بودن این کنترل، ویژگی‌های بسیار جالبی در آن موجود است. به عنوان مثال می‌توان در سه حالت عدد صحیح از ۰ تا ۹۹، به عنوان عدد اعشاری و حتی به عنوان متنی، لیستی از مقادیر را بپذیرد. در زیر برای هر یک از آن‌ها مثالی آورده شده است که به جزئیات و نحوه عملکرد آن‌ها می‌پردازیم:

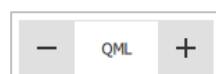
```
SpinBox {
    from: 0
    to: items.length - 1
    value: 1 // "QML"

    property var items: ["C++", "QML", "JavaScript"]

    textFromValue: function(value) {
        return items[value];
    }
}
```

در ابتدای کد مقدار ابتدایی آن برابر ۰ قرار گرفته است، همچنین سقف مقدار تعریف شده آن بر پایه تعداد آیتم‌هایی تعیین شده است که در متغیر items آمده است. این متغیر شامل سه مقدار C++, QML, JavaScript است. جهت ارسال مقادیر مورد نظر تحت جاوااسکریپت می‌باشد از صفت textFromValue با ترکیب تابع بازگشت دهنده مقدار لیست استفاده شود.

در این بخش تابع وظیفه دریافت مقدار ثابتی را دارد که به عنوان شناسه صحیح یا همان ایندکس آیتم‌های موجود در متغیر items است.



در ادامه مثال فوق مربوط به مقدار صحیحی است که در بازه ۰ تا ۱۰۰ تعریف شده است.

```
SpinBox {
    id: spinbox
    from: 0
    value: 10
    to: 100
    stepSize: 10
    anchors.centerIn: parent

    property int decimals: 2

    textFromValue: function(value, locale)
    {
        return Number(value / 100).toLocaleString
            (locale, 'f', spinbox.decimals)
    }
}
```

}

همانطور که مشخص است بازه‌این مثال بین ۰ تا ۱۰۰ مشخص شده است، همچنین صفت `stepSize` به عنوان تعداد مقادیر برای جهش یا پرش بر روی مقدار است برای مثال با کلیک بر روی اضافه کننده (+) مقدار ۰.۱۰ به ۰.۲۰ تغییر خواهد یافت این به دلیل همین خاصیت `stepSize` که ۱۰ مقدار یکجا پرش پیدا می‌کند. در ادامه با تعریف متغیر `decimals` از نوع عدد صحیح مقادیر فوق را توسط تابع `javaScript` دریافت و تحت خاصیت `textFromValue` اعمال می‌کند.

البته توجه داشته باشید تابع مذکور با دریافت دو آرگومان از نوع `value` و `locale` وظیفه تبدیل قالب بازگشتی را بر عهده دارد. البته آرگومان دوم اختیاری است. که بر پایه `Qt.locale()` جهت تشخیص مناسب‌ترین قالب بر اساس محل و موقعیت زمانی تاثیر گذار است. که در رابطه با آن در بخش بحث `Qt.Local` توضیحات کامل ارائه خواهد شد.



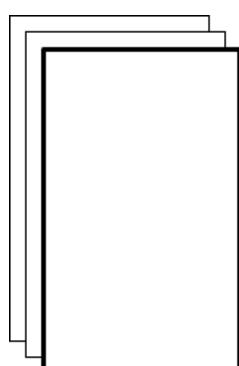
اما این فرمت چه کاربردی دارد؟ در یک مثال بسیار ساده می‌توان به آن اشاره کرد که در صورت نیاز بر اساس محلی که مشخص می‌کنید فرمت تبدیل شده سفارشی سازی خواهد شد. تحت کد (`Qt.locale("fa_IR")` می‌توان به راحتی نوع خروجی را با جایگزین بر `locale` سفارشی سازی کرد.



## convertView کنترل

این کنترل به عنوان نمای پشتی (Stack-View) است - یکی از بهترین و کاربردی‌ترین کنترل‌های موجود در رابطه‌ای کاربری مدرن و مخصوصاً تحت موبایل محسوب می‌شود. لذا می‌توان توسط آن مجموعه‌ای از صفحات را جمع آوری کرده و از اطلاعات آنها استفاده کرد. به عنوان نمونه، فرض کنید در محیط ارسال یک پیغام هستید. در این بخش در زمان وارد کردن متن مورد نظر، نیاز است در آن واحد وارد یک پنجره دیگری شده و یک فایلی را به عنوان ضمیمه وارد متن خود کنید. این کنترل با در اختیار گذاشتن مجموعه‌ای از صفحات مورد نظر شما قابلیت سوئیچ بین آنها را فراهم می‌کند.

شکل کلی کد و خروجی این کنترل به صورت زیر است:



`StackView {`

```

    id: stack
    initialItem: mainView
    anchors.fill: parent
}

Component {
    id: mainView

    Row {
        spacing: 10

        Button {
            text: "Push"
            onClicked: stack.push(mainView)
        }
        Button {
            text: "Pop"
            enabled: stack.depth > 1
            onClicked: stack.pop()
        }
        Text {
            text: stack.depth
        }
    }
}

```

نوع	صفت
bool	busy
item	currentItem
int	depth
var	initialItem
Transition	from
Transition	popEnter
Transition	popExit
Transition	pushEnter
Transition	pushExit
Transition	replaceEnter
Transition	replaceExit
توضیح عملکرد سیگنال	سیگنال
سیگنال متصل شده زمانی ساطع می‌شود که آیتم، در استک متصل فعال شده باشد.	activated
سیگنال متصل شده زمانی ساطع می‌شود که آیتم، در استک متصل در حالت فعال شدن باشد.	activating
سیگنال متصل شده زمانی ساطع می‌شود که آیتم، در استک متصل در حالت غیرفعال باشد.	deactivated
سیگنال متصل شده زمانی ساطع می‌شود که آیتم در استک متصل در حالت غیرفعال شدن باشد.	deactivating
زمانی این سیگنال ساطع می‌شود که آیتم موجود در استک خذف شده باشد.	removed
توضیح عملکرد تابع	عملکرد
همه آیتم‌های موجود در استک را حذف می‌کند. هیچ اینیمیشنی در این عملکرد وجود نخواهد داشت.	clear
جستجوی آیتم مورد نظر در استک.	find
موقعیت یک آیتم را در استک بر می‌گرداند، در صورتی که بدون نتیجه باشد مقدار خالی "null" را بر می‌گرداند.	get
یک یا چند آیتم غیرفعال شده در استک را به بالاترین سطح می‌آورد. آخرین آیتم حذف شده موجود در استک را بر می‌گرداند.	pop
آیتم را برای استفاده مشخصی در داخل استک هُل می‌دهد.	push
یک یا چند آیتم را در استک برای آیتم و عمل مشخصی جایگزین می‌کند.	replace

استفاده از این کنترل به اندازه‌ای به کمک راحتی کار می‌آید که به شما اجازه می‌دهد به سادگی یک زیر مجموع (فرزنده) را برای پنجره اصلی که از نوع (والد) محسوب می‌شود اضافه و آن‌ها را مدیریت کنید. استک معمولاً با خاصیت‌های لنگر اندازی (anchors) به هر چهار سوی والد خود می‌تواند ادغام شود.

کنترل نمایه پشت‌هه (Stack View) از سه عمل اصلی پشتیبانی می‌کند که عبارتند از push(), pop(), replace(): این عملیات با عملیات پشت‌هه کلاسیک مطابقت می‌کند که در آن push یک آیتم به بالای پشت‌هه اضافه می‌کند و pop بالاترین آیتم را از پشت‌هه حذف می‌کند و هر جایی که "push" به عنوان یک آیتم در بالاترین سطح پشت‌هه - استک (Stack) قرار می‌گیرد، "pop" بالاترین آیتم را از استک حذف می‌کند، و عمل replace هم مانند pop بوده و توسط push دنبال می‌شود، که آیتم جدید با کدام آیتم که اغلب در بالای پشت‌هه قرار دارد جایگزین شود. بالاترین آیتمی که در پشت‌هه قرار می‌گیرد همان چیزی است که در حال حاضر در صفحه نمایش نمایان می‌شود.

توجه: وقتی استک خالی باشد، یک push نمی‌تواند خاصیت انتقال حالت را همراه با اینیمیشن داشته باشد. زیرا چیزی وجود ندارد که به آن انتقال پیدا کند.

گاهی اوقات، لازم است که بیشتر از یک قدم به عقب در استک رجوع کنیم. برای مثال، برای بازگشت به یک آیتم "اصلی" یا برخی از آیتم‌های برنامه. در اینگونه موارد ممکن است که یک آیتم را به عنوان یک پارامتر برای pop() مشخص کند. این عمل "unwind" نامیده می‌شود.

### یافتن آیتم در استک

کد زیر وظیفه جستجو یک آیتم را در استک توسط شناسه "order\_id" دارد.

```
stack.pop(stack.find(function(item) {  
    return item.name == "order_id";  
}));
```

همچنین شما می‌توانید با استفاده از get(index) آیتمی را از استک دریافت کنید.

```
previousItem = stack.get(myItem.stack.index - 1));
```

### انتقال - تغییر حالت

برای هر کدام از عملیات pop یا push از اینیمیشن‌های مختلفی استفاده می‌شود. این اینیمیشن‌ها مرتبط با حالت خروج و ورود به آیتم‌ها هستند. اینیمیشن می‌تواند برای StackView replaceExit، replaceEnter، popExit، popEnter، pushExit، pushEnter سفارشی سازی شود.

تکه کد زیر نمونه‌ای از اینیمیشن با حالت "محو شدن" برای عملیات push و pop تعریف شده است:

```
StackView {  
  id: stackview  
  anchors.fill: parent  
  
  pushEnter: Transition {
```

```

        PropertyAnimation {
            property: "opacity"
            from: 0
            to:1
            duration: 200
        }
    }
    pushExit: Transition {
        PropertyAnimation {
            property: "opacity"
            from: 1
            to:0
            duration: 200
        }
    }
}

popEnter: Transition {
    PropertyAnimation {
        property: "opacity"
        from: 0
        to:1
        duration: 200
    }
}
popExit: Transition {
    PropertyAnimation {
        property: "opacity"
        from: 1
        to:0
        duration: 200
    }
}
}
}

```

همانطور که مشخص است در کد بالا صفت‌های `Transition` همراه با خاصیت `StackView` ترکیب شده‌اند. یک `Animation` اینیمیشن‌هایی را برای انواع حالت‌های مختلف (وضعیت‌های متفاوت) در یک کنترل یا شیء‌ای که رخ می‌دهد را ارائه می‌کند. همچنین `PropertyAnimation` روشی را برای تغییر حالت همراه با مقادیر مرتبط با اینیمیشن فراهم می‌کند که به توضیحات جزئی پرداخته خواهد شد.

## کنترل `SwipeDelegate`

کنترل `SwipeDelegate` نمایی از آیتم را فراهم می‌کند که می‌تواند به سمت راست و چپ کشیده شود تا اطلاعات بیشتری را در رابطه با آیتم مربوطه ارائه دهد. درواقع توضیحات تا زمانی که آیتم کشیده نشود مخفی خواهند بود. این کنترل به عنوان نماینده با کنترل `ListView` مورد استفاده قرار می‌گیرد. به لیست تماس‌های گوشی‌های موبایل خود توجه کنید، خواهید دید که با کشیدن روی یکی از تماس‌ها توضیحات و یا دکمه‌ای به عنوان `Delete` مشاهده خواهید کرد. این قابلیت دقیقاً همان `SwipeDelegate` است که در لیست تماس‌ها موجود است.

صفت	نوع
<code>swipe.position</code>	real
<code>swipe.complete</code>	bool
<code>swipe.left</code>	Component
<code>swipe.behind</code>	Component

<b>swipe.right</b>	Component
<b>swipe.leftItem</b>	Item
<b>swipe.behindItem</b>	Item
<b>swipe.rightItem</b>	Item

این کنترل رابطه‌ای برنامه‌نویسی نوع **AbstractButton** را به ارث برده است. کد زیر نمایی از خود کنترل است و به تنها‌یی مورد استفاده قرار گرفته است:

```
SwipeDelegate {
    id: swipeDelegate
    text: "Kambiz Asadzadeh"
    width: parent.width
    swipe.right: Label {
        id: deleteLabel
        text: qsTr("Delete")
        color: "white"
        verticalAlignment: Label.AlignVCenter
        padding: 12
        height: parent.height
        anchors.right: parent.right
        background: Rectangle {
            color: 'red'
        }
    }
}
```

همانطور که مشخص است در کد فوق با استفاده از کنترل ذکر شده متن مورد نظر مشخص شده و سپس با استفاده از صفت **swipe.right** کنترل مورد نظر را که در اینجا یک برچسب (**Label**) است وارد می‌کنیم. توجه داشته باشید کدهای اضافی مرتبط با سفارشی سازی بوده و اطلاعات کافی در زمینه سفارشی سازی کنترل‌ها ارائه می‌شود که نتیجه نهایی کد مربوطه به شکل زیر خواهد بود:



بهترین کاربرد این کنترل زمانی است که لازم باشد از یک لیست استفاده کنید، بنابراین اگر می‌خواهید فرمی مانند لیست تماس گوشی یا لیست دوستان خود در نرم‌افزاری مانند تلگرام طراحی کنید، در این صورت این گزینه یکی از بهترین موارد است که می‌توان در سفارشی سازی لیست ها مفید باشد. بنابراین ترکیب یک لیست با این کنترل به صورت زیر خواهد بود:

```
ListView {
    id: listView

    anchors.fill: parent
    model: ListModel {
        ListElement { title: "C/C++"; }
        ListElement { title: "QML/JavaScript"; }
        ListElement { title: "HTML5/CSS3"; }
    }
    delegate: SwipeDelegate {
        id: swipeDelegate
```

```

text: model.title
width: parent.width
ListView.onRemove: SequentialAnimation {
    PropertyAction {
        target: swipeDelegate
        property: "ListView.delayRemove"
        value: true
    }
    NumberAnimation {
        target: swipeDelegate
        property: "height"
        to: 0
        easing.type: Easing.InOutQuad
    }
    PropertyAction {
        target: swipeDelegate;
        property: "ListView.delayRemove";
        value: false
    }
}
swipe.right: Label {
    id: deleteLabel
    text: qsTr("Delete")
    color: "white"
    verticalAlignment: Label.AlignVCenter
    padding: 12
    height: parent.height
    anchors.right: parent.right
    SwipeDelegate.onClicked: listView.model.remove(index)

    background: Rectangle {
        color: "#ff6347"
    }
}
}
}

```

استفاده از کنترل `SwipeDelegate` همراه با یک کنترل نگه دارنده لیست، مانند `ListView` بسیار کاربردی است. تنها نکته‌ای که باید به آن توجه داشته باشید این است که در ترکیب کنترل‌ها به یکدیگر صفت `delegate` با حرف کوچک نوشته می‌شود. این قانون در تمامی کنترل‌های موجود در کنترل‌های کیوت کوئیک صدق می‌کند.

در داخل بدن کنترل صفت `text` از حالت عادی خارج شده و به صورت پذیرنده یک لیست توسط کنترل `ListElement` مدیریت می‌شود که با مقدار دهی کردن مشخصه `model` می‌توان به عنوان‌های آیتم‌ها دسترسی داشت. در رابطه با `ListElement` توضیحات در صفحه مورد نظر ارائه خواهد شد.

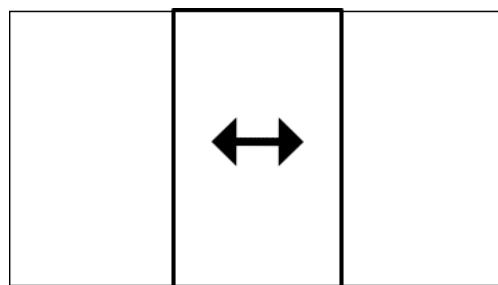
## کنترل `SwipeView`

این کنترل یک منو، یا سیستم ناوبری بر پایه حالت کشیده شدن را فراهم می‌کند. در این کنترل مجموعه‌ای از صفحات جمع آوری می‌شود. هر صفحه در زمان معینی قابل نمایش است. کاربر می‌تواند بین صفحات به صورت لمسی کشیده و حرکت کند. توجه داشته باشید که خود کنترل

کاملاً غیر بصری است بنابراین توصیه می‌شود برای استفاده هر چه بهتر آن را با کنترل `PageIndicator` ترکیب کنید تا نشانی از جلوه‌های بصری در این کنترل نمایان شود.

صفت	نوع
<code>interactive</code>	<code>bool</code>

شکل کلی کنترل و کد مرتبط با آن به صورت زیر است:



```
SwipeView {  
    id: view  
    currentIndex: 1  
    anchors.fill: parent  
  
    Item {  
        id: firstPage  
    }  
    Item {  
        id: secondPage  
    }  
    Item {  
        id: thirdPage  
    }  
}
```

در نهایت برای ترکیب کنترل با یک `PageIndicator` می‌بایست به صورت زیر عمل کنید:

```
SwipeView {  
    id: view  
    currentIndex: 1  
    anchors.fill: parent
```

```

Item {
    id: firstPage
    Rectangle {
        color: 'red'
        anchors.fill: parent
        Text {
            text: qsTr("C++")
            anchors.centerIn: parent
            color: 'white'
            font.pixelSize: 32
        }
    }
}
Item {
    id: secondPage
    Rectangle {
        color: 'green'
        anchors.fill: parent
        Text {
            text: qsTr("QML")
            anchors.centerIn: parent
            color: 'white'
            font.pixelSize: 32
        }
    }
}
Item {
    id: thirdPage
    Rectangle {

        color: 'blue'
        anchors.fill: parent
        Text {
            text: qsTr("HTML5")
            anchors.centerIn: parent
            color: 'white'
            font.pixelSize: 32
        }
    }
}
}

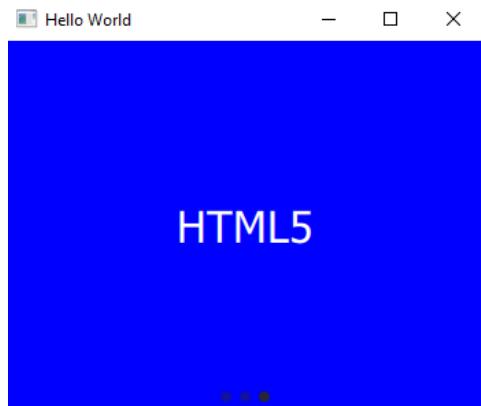
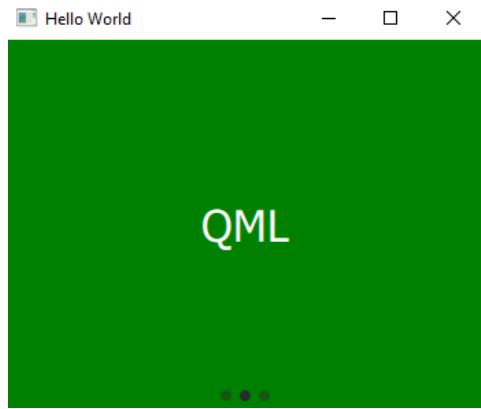
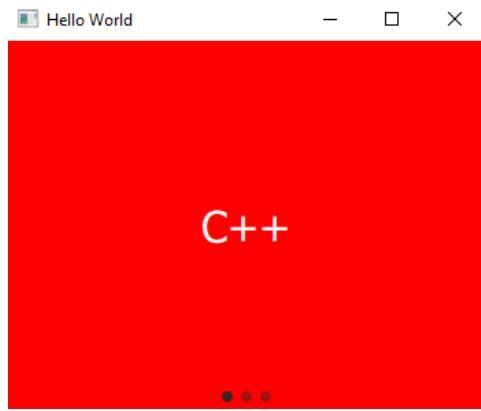
PageIndicator {
    id: indicator

    count: view.count
    currentIndex: view.currentIndex

    anchors.bottom: view.bottom
    anchors.horizontalCenter: parent.horizontalCenter
}

```

در کد ذکر شده کنترل `SwipeView` تعریف و سپس در داخل بدن آن اقدام به تعریف آیتم‌های مورد نظر کرده‌ایم که هر یک از آن‌ها شامل اشیاء فرزند هستند. سپس در نهایت کنترل `PageIndicator` تعریف و خاصیت `count` آن برابر با شناسه کنترل والد یعنی `SwipeView` شده است. حال با تعیین `currentIndex` به عنوان دریافت کننده‌ایندکس جاری از طرف کنترل والد تعیین شده است. در رابطه با `anchors` ها توضیحات در بخش مرتبط با سفارشی سازی داده شده است که نتیجه کد فوق به صورت زیر خواهد بود:



جهت سوئیچ بین آیتم‌ها کافی است بر روی صفحه عمل کشیدن را انجام دهید.

### کنترل **Switch**

کنترل سوئیچ، یک کنترل به عنوان دکمهٔ به شیوهٔ انتخاب گزینه‌ای است که با کشیده شدن و جابجا شدن (فعال) و (غیرفعال) می‌شود. معمولاً این کلید برای استفاده بین دو حالت فعال و غیرفعال استفاده می‌شود.

صفت	نوع
<b>position</b>	real
<b>visualPosition</b>	real

شكل کلی کنترل و کد مرتبط با آن به صورت زیر است:

```
ColumnLayout {
```

```
    Switch {
```

```

    text: qsTr("C++")
}

Switch {
    text: qsTr("QML")
}

```



با توجه به یکی از رویدادهای این کنترل به نام `onCheckedChanged` می‌توان کدی به صورت زیر را فراهم نمود:

```

ColumnLayout {
    Switch {
        text: qsTr("Do you like C++ programming language?")
        onCheckedChanged: {
            if(!checked){
                checkstatus.text = "NO !";
            } else {checkstatus.text = "Yes!"}
        }
    }

    Label {
        id:checkstatus
        text: 'Status'
    }
}

```

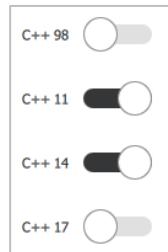
در کد ذکر شده در داخل خاصیت `onCheckedChanged` با استفاده از یک دستور شرطی ساده وضعیت متن کنترل `Label` را هنگام تغییر وضعیت کنترل `CheckBox` تغییر می‌دهیم. این دستور با بررسی وضعیت انتخابی با `checked` پاسخ می‌دهد.

## کنترل `SwitchDelegate`

همانطور که انتظار می‌رود این کنترل به عنوان یک نماینده از کنترل قبلی یعنی `Switch` جهت ترکیب و استفاده در صفت `delegate` لیست `(ListView)` و دیگر کنترل‌ها مورد استفاده قرار می‌گیرد.

ساختار این کنترل نیز به صورت زیر است:

```
ListView {  
    model: ["C++", "QML", "HTML5"]  
    delegate: SwitchDelegate {  
        text: modelData  
    }  
}
```



در نهایت کد نهایی به صورت زیر خواهد بود:

```
ListView {  
    anchors.fill: parent  
    model: ["C++ 98", "C++ 11", "C++ 14", "C++ 17"]  
    delegate: SwitchDelegate {  
        text: modelData  
    }  
}
```

تحت کنترل `ListView` و صفت `model` لیستی از آیتم‌های مورد نظر را تعریف نموده و سپس در صفت `delegate` از کنترل `SwitchDelegate` استفاده می‌کنیم. توجه کنید که مقدار صفت `text` این کنترل برابر است با `modelData` که نقش ارائه دهنده مدل‌های داده را بر عهده دارد.

## کنترل `TabBar`

این کنترل نیز یک مدلی از منو یا همان ناوبری است که به عنوان نوار یا زبانه ظاهر می‌شود.  
مدل و شکل کلی آن به صورت زیر است:

```
TabBar {  
    id: bar  
    width: parent.width  
    TabButton {  
        text: qsTr("Home")  
    }  
    TabButton {  
        text: qsTr("About")  
    }  
}
```

```

    text: qsTr("News")
}

TabButton {
    text: qsTr("Library")
}

}

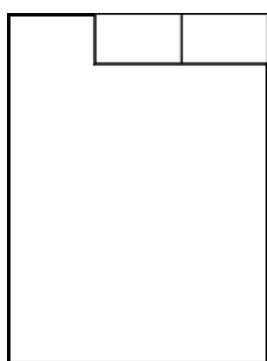
StackLayout {
    width: parent.width
    currentIndex: bar.currentIndex

    Item {
        id: homeTab
    }

    Item {
        id: newsTab
    }

    Item {
        id: libraryTab
    }
}

```



توجه داشته باشید که عنوان آیتم‌های مربوط به این کنترل توسط کنترل فرزند آن یعنی `TabButton` و محتوای آن‌ها تحت کنترل لایه بندی `StackLayout` فراهم می‌شود.

توجه جهت استفاده از `StackLayout` می‌بایست از ماتریس زیر استفاده کنید:

```
import QtQuick.Layouts 1.3
```

کد نهایی به عنوان یک نمونه به صورت زیر است :

```
import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.3

ApplicationWindow {
    id:window
    width: 800
    height: 600
    visible: true
    title: qsTr("Hello World")

    TabBar {
        id: bar
        width: parent.width
        TabButton {
            text: qsTr("Home")
        }
        TabButton {
            text: qsTr("News")
        }
        TabButton {
            text: qsTr("Library")
        }
    }

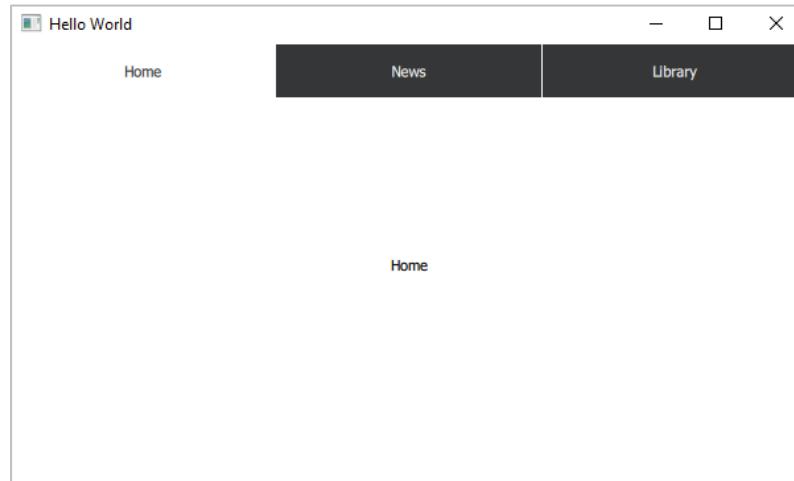
    StackLayout {
        width: parent.width
        currentIndex: bar.currentIndex

        anchors.fill: parent

        Item {
            id: homeTab
            Text {
                anchors.centerIn: parent
                text: "Home"
            }
        }
        Item {
            id: newsTab
            Text {
                anchors.centerIn: parent
                text: "News"
            }
        }
        Item {
            id: libraryTab
            Text {
                anchors.centerIn: parent
                text: "Library"
            }
        }
    }
}
```

}

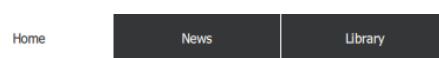
همچنین خروجی آن به شکل زیر خواهد بود که می‌توانید آن را سفارشی سازی نمایید:



## TabButton کنترل

طبق روال این کنترل به عنوان یک کنترل فرزند می‌توان تحت مدیریت کنترل والد خود یعنی `TabBar` مورد استفاده قرار گیرد و به عنوان دکمه‌ای عناوین مرتبط با آن نمایان می‌شوند که شکل کلی آن به صورت زیر است:

```
TabBar {  
    id: bar  
    width: parent.width  
  
    TabButton {  
        text: qsTr("Home")  
    }  
  
    TabButton {  
        text: qsTr("News")  
    }  
  
    TabButton {  
        text: qsTr("Library")  
    }  
}
```



## TextArea کنترل

کنترل تکست اریا (TextArea) یا همان ناحیه متنی با قابلیت ویرایش متن با پشتیبانی از چند سطر. این کنترل مدل گسترش یافته است که همراه با خاصیت متن نمونه (placeholder) جهت افزایش دکوراسیون فراهم شده است.

```
TextArea {  
    placeholderText: qsTr("Enter your biography")  
}
```

TextArea  
...  
...  
...

نوع	صفت
item	background
enumeration	focusReason
bool	hoverEnabled
bool	hovered
string	placeholderText
توضیح عملکرد سیگنال	
این سیگنال زمانی ساطع می‌شود که به مدت طولانی فشار داده شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	pressAndHold
این سیگنال زمانی ساطع خواهد شد که منطقه مربوط به متن توسط کاربر فشرده شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	pressed
این سیگنال زمانی ساطع خواهد شد که منطقه مربوط به متن توسط کاربر رها شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	released

این کنترل می‌تواند با لایه‌های دیگری همچون Flickable ترکیب شود.

```
Flickable {  
    id: flickable  
    anchors.fill: parent  
  
    TextArea.flickable: TextArea {  
        text: "My text\n ... \n... \n.... \n"  
        wrapMode: TextArea.Wrap  
    }  
  
    ScrollBar.vertical: ScrollBar { }  
}
```

در کد فوق با استفاده از یک کنترل Flickable با تعیین کنترل TextArea به عنوان شیء قابل مدیریت تحت کنترل والد صورت گرفته است. همچنین خاصیت wrapMode در این کنترل قابلیت پشتیبانی از چند خط را فراهم می‌کند.

## TextField کنترل

این کنترل همانند کنترل (TextArea) قابلیت ویرایش متن را با پشتیبانی از یک سطر را دارد. این کنترل مدل گسترش یافته (placeholder) جهت افزایش دکوراسیون فراهم شده است.

```
TextField {
    placeholderText: qsTr("Enter your Name")
}
```

نوع	صفت
item	background
enumeration	focusReason
bool	hoverEnabled
bool	hovered
string	placeholderText
توضیح عملکرد سیگنال	سیگنال
این سیگنال زمانی ساطع می‌شود که به مدت طولانی فشار داده شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	pressAndHold
این سیگنال زمانی ساطع خواهد شد که منطقه مربوط به متن توسط کاربر فشرده شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	pressed
این سیگنال زمانی ساطع خواهد شد که منطقه مربوط به متن توسط کاربر رها شود. البته مقدار زمان بستگی به نوع پلتفرم دارد. این رویداد پارامترهایی را که شامل اطلاعاتی در مورد مختصات X و Y است را خواهد داد.	released

کد زیر یک نمونه قابل استفاده از این کنترل را فراهم می‌کند:

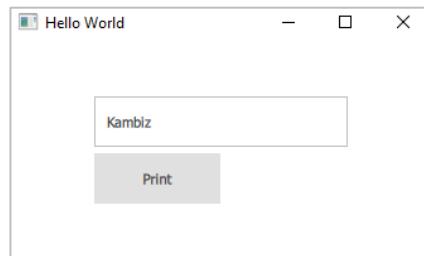
```
ColumnLayout {
    anchors.centerIn: parent

    TextField {
        id:firstname
        placeholderText: qsTr("Enter your Name")
    }

    Button {
        id:buttonPrint
        text: "Print"
        onClicked: {
            console.log("Your name is: " + firstname.text)
        }
    }
}
```

در کد مذکور با استفاده از یک دکمه **Button** مقدار وارد شده بر کنترل **TextField** را چاپ می‌کنیم که نتیجه خروجی و طرح آن به شکل زیر خواهد بود:

qml: Your name is: Kambiz



## كنترل ToolBar

این کنترل یک نگهدارنده یا ظرفی است که به طور گسترده بخش حساس به عملیات و ابزارهای کنترلی را مانند، منو، دکمه‌ها و فیلد‌های جستجو کنترل می‌کند. این کنترل هیچ لایه‌ای را از خود نمی‌سازد، بنابراین در صورتی که نیاز باشد موقعیت محتوای درج شد در آن را سفارشی کنید می‌بایست از کنترل‌های لایه‌ای مانند **RowLayout** استفاده کنید. اندازه آیتم تحت این روش می‌تواند بسیار مناسب و مدیریت شده تعیین شود.

صفت	نوع
<b>Position</b>	enumeration

کد زیر یک نمونه قابل استفاده از این کنترل را فراهم می‌کند:

```
ApplicationWindow {
    id:window
    width: 800
    height: 600
    visible: true
    title: qsTr("Hello World")

    header: ToolBar {
        RowLayout {
            anchors.fill: parent

            ToolButton {
                text: qsTr("\u25C0 %1").arg(Qt.application.name)
                enabled: stack.depth > 1
                onClicked: stack.pop()
            }

            Item { Layout.fillWidth: true }

            Switch {
                checked: true
                text: qsTr("Notification")
            }
        }
    }
}
```

```

StackView {
    id: stack
    anchors.fill: parent
}

}

```

کنترل **ToolBar** به عنوان یک شیء فرزند تحت مدیریت صفت **header** در کنترل والد اصلی **ApplicationWindow** مورد استفاده قرار می‌گیرد که کد فوق یک نمونه سفارشی شده از آن محسوب می‌شود. در این میان ترکیب کنترل استک و استفاده از قابلیت‌های آن ترکیب بسیار کاربردی را فراهم ساخته است.

## کنترل **ToolButton**

این کنترل نیز به عنوان کنترل مکمل برای **ToolBar** در نظر گرفته شده است. خصوصیات عملیاتی آن شباهت زیادی به کنترل **Button** دارد و می‌توان از خصیت‌های آن به عنوان انجام فراخوانی از یک تابع یا هر عمل دیگری را استفاده کرد.

کد زیر یک نمونه قابل استفاده از این کنترل را فراهم می‌کند:

```

ToolBar {
    RowLayout {
        anchors.fill: parent
        ToolButton {
            text: qsTr("\u25C0 My Application")
            onClicked: stack.pop()
        }
        Item { Layout.fillWidth: true }
        Switch {
            checked: true
            text: qsTr("Do action")
        }
    }
}

```

## کنترل **ToolSeparator**

کنترل **ToolSeparator** همانطور که از نامش مشخص است یک جلوه بصری در بین دو گروه از کنترل در نوار ابزار قرار می‌گیرد. این کنترل توسط یک خط کنترل‌های گروه بندی شده را از یکدیگر جدا می‌سازد که با خاصیت عمودی و افقی انجام می‌پذیرد.

صفت	نوع
<b>horizontal</b>	bool
<b>orientation</b>	enumeration
<b>vertical</b>	bool

شکل کلی این کنترل به صورت زیر است:

```
ToolBar {
```



کد زیر یک نمونه قابل استفاده از این کنترل را فراهم می‌کند:

```
ToolBar {
```

```
    RowLayout {
        anchors.fill: parent

        ToolButton {
            text: qsTr("Action 1")
        }
        ToolButton {
            text: qsTr("Action 2")
        }

        ToolSeparator {}

        ToolButton {
            text: qsTr("Action 3")
        }
        ToolButton {
            text: qsTr("Action 4")
        }

        ToolSeparator {}

        ToolButton {
            text: qsTr("Action 5")
        }
```

```
    ToolButton {
        text: qsTr("Action 6")
    }
```

```
    Item {
        Layout.fillWidth: true
    }
}
```

کد مربوطه نشانگر این است که در یک کنترل های موجود که در یک ردیف توسط کنترل لایه `RowLayout` مدیریت می‌شوند صورت می‌گیرد.

کنترل `ToolTip`

کنترل ابزار (ToolTip) یکی از معروف‌ترین ابزارهایی است که در طراحی مورد استفاده قرار می‌گیرد. این کنترل متنی را دریافت و آن را برای نمایش اطلاعات بیشتر در اختیار کاربر قرار می‌دهد.

صفت	نوع
delay	int
text	string
timeout	int

این کنترل به صورت زیر والد و از نوع یک شیء به صورت فرزند عمل می‌کند و به تنها ی کاربردی ندارد. جهت استفاده از آن می‌بایست از کنترل استفاده شود که خاصیت فشرده شدن یا عملیات مشابه را داشته باشد. برای مثال در زیر تحت خاصیت فشرده شدن یک کنترل Button به آن اشاره می‌کنیم:

```
Button {
    text: qsTr("Button")
    id:button
    hoverEnabled: true
    ToolTip.delay: 1000
    ToolTip.timeout: 5000
    ToolTip.visible: button.pressed
    ToolTip.text: qsTr("بعد از یک ثانیه این پیغام")
    " با حرکت اشاره گر بر روی کنترل نمایان خواهد شد.
}
```

به عنوان مثال در کد بالا با استفاده از کنترل دکمه Button با فعال‌سازی صفت‌های hovedEnabled جهت فعال‌سازی پذیرش قابلیت شناور شدن ماوس و همچنین خاصیت delay جهت تخصیص زمان تاخیر در نمایش به مقدار ۱۰۰۰ میلی ثانیه است. با درج این مقدار در مدت ۱ ثانیه طول خواهد کشید تا متن مورد نظر به عنوان یک Tool Tip نمایش داده شود. البته خاصیت timeout نیز برای مدت زمان قابل نمایش در نظر گرفته شده است.

خاصیت visible این کنترل برابر hovered که در صورت حرکت ماوس بر روی کنترل مادر رویداد نمایش این کنترل اتفاق خواهد افتاد. توجه داشته باشید که خاصیت visible برای این کنترل می‌تواند بر اساس رویدادهای شیء والد تغییر کند. برای مثال می‌توانید با درج کد عمل نمایش را به رویداد کلیک شدن منتقل نمایید.

## کنترل Tumbler

کنترل ثامبلر (Thumblter) اجازه می‌دهد تا کاربر گزینه‌ای را در حالت چرخ دادن به شیء مورد نظر انتخاب کند. کنترلی شبیه صفحه رمزی قاوصندوق‌ها یا کیف‌های همراه! زمانی این کنترل بسیار مفید است که نیاز باشد از چند کنترل مانند SpinBox و RadioButton در کنار یکدیگر استفاده شود، در این حالت جایگزین کنترل Tumbler بسیار کار آمد خواهد بود.

صفت	نوع
-----	-----

<b>count</b>	int
<b>currentIndex</b>	int
<b>currentItem</b>	item
<b>delegate</b>	Component
<b>model</b>	variant
<b>visibleItemCount</b>	int
<b>wrap</b>	bool

رابطهای برنامه‌نویسی این کنترل بیشتر شبیه به کنترل‌های لیست کننده مانند `PathView` و `ListView` است. این کنترل می‌تواند به عنوان یک نماینده از کنترل اصلی تحت خاصیت `delegate` کنترل‌های فوق مورد استفاده قرار بگیرد.

دقیق کنید که مشابه این کنترل در ماژول Extra نیز موجود است، اما این نسخه مختص ماژول Qt Quick Controls تعبیه گردیده است تا از خاصیت‌های ویژه کیوت ۵.۷ و ۵.۸ به بالا را پشتیبانی کند.

شكل کلی این کنترل به صورت زیر است:

```
Tumbler {
    id: control
    model: 5
    anchors.centerIn: parent
}
```

خاصیت `model` همانطور که مشخص است می‌تواند با پذیرش قابلیت شیء فرزند به صورت بسیار زیبایی سفارشی سازی شود.

```
Tumbler {
    id: control
    model: 5
    anchors.centerIn: parent

    background: Item {
        Rectangle {
            opacity: control.enabled ? 0.2 : 0.1
            border.color: "#000000"
            width: parent.width
            height: 1
            anchors.top: parent.top
        }

        Rectangle {
            opacity: control.enabled ? 0.2 : 0.1
            border.color: "#000000"
            width: parent.width
            height: 1
            anchors.bottom: parent.bottom
        }
    }

    delegate: Text {
        text: qsTr("Number %1").arg(modelData + 1)
        font: control.font
        horizontalAlignment: Text.AlignHCenter
        verticalAlignment: Text.AlignVCenter
    }
}
```

```

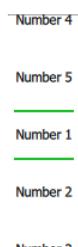
Rectangle {
    anchors.horizontalCenter: control.horizontalCenter
    y: control.height * 0.4
    width: 50
    height: 2
    color: "#21be2b"
}

Rectangle {
    anchors.horizontalCenter: control.horizontalCenter
    y: control.height * 0.6
    width: 50
    height: 2
    color: "#21be2b"
}
}

```

در کد مذکور با تعریف مقدار ۵ آیتم به صفت `model` در کنترل تعداد آیتم‌های موجود را مشخص کرده‌ایم. البته توجه داشته باشید همانطور که ذکر شد امکان سفارشی سازی مدل داده‌ای از طرف C++ و JavaScript ممکن است که به آن‌ها اشاره خواهد شد. بنابراین مقدار ۵ در این آموزش به عنوان یک مقدار آزمایشی ذکر شده است.

در ادامه با سفارشی سازی صفت‌های `background` و `delegate` یک صفحه سفارشی را نمایان خواهد کرد که نتیجه آن به صورت زیر خواهد بود:



## فصل ششم

### معرفی Qt Quick Dialog

#### Dialog کنترل

کنترل Dialog نوعی پنجره مناسب برای گفتگو با کاربر محتوای دلخواه شما را در قالب یک پنجره نمایش می‌دهد. این کنترل دارای مشخصه `contentItem` است که اجازه دریافت محتوا به عنوان فرزند را فراهم می‌کند. که می‌تواند شامل کنترل‌های استاندارد باشد.

جهت استفاده از این کنترل باید از ماژول مرتبط آن استفاده شود که به صورت زیر است:

```
import QtQuick.Dialogs 1.2
```

این کنترل مانند کنترل‌های اصلی [Page](#) و [ApplicationWindow](#) پنجره‌ای را فراهم می‌کند که اجازه می‌دهد در بین پنجره‌های موجود فعال شود. همچنین این کنترل قابلیت سفارشی سازی دکمه‌های مورد عمل را دارد.

نوع	صفت
QPushButton	<a href="#">clickedButton</a>
QObject	<a href="#">contentItem</a>
item	<a href="#">modality</a>
StandardButtons	<a href="#">standardButtons</a>
string	<a href="#">title</a>
bool	<a href="#">visible</a>
توضیح عملکرد سیگنال	سیگنال
سیگنال ساطع شده زمانی خواهد بود که خاصیت رویداد accept آن اجرا و پذیرفته شود.	<a href="#">accepted</a>
سیگنال ساطع شده زمانی خواهد بود که خاصیت رویداد reject آن اجرا و پذیرفته شود. و یا برخلاف آن سیگنال accept مورد پذیرش قرار نگیرد.	<a href="#">rejected</a>
توضیح عملکرد تابع	عملکرد
پنجره مرتبط با کنترل را بسته و سپس سیگنال پذیرش را ارسال می‌کند.	<a href="#">accept</a>
پنجره مرتبط با کنترل را بسته و سپس سیگنال عدم پذیرش را ارسال می‌کند.	<a href="#">reject</a>

کد زیر نحوه استفاده از این کنترل را نمایش می‌دهد که نتیجه آن طبق تصویر بعدی خواهد بود:

```
Dialog {
    visible: true
    title: "Greeting"

    onAccepted: console.log("OK!")()
    contentItem: Rectangle {
        color: "white"
        implicitWidth: 400
        implicitHeight: 100
        Text {
            text: "Hello, This is Me!"
            color: "black"
            anchors.centerIn: parent
        }
    }
}
```



در ادامه لیستی از دیگر ثابت‌های این کنترل را مشاهده می‌کنید که اجازه می‌دهند بر اساس ثابت‌های از قبل تعریف شده پنجره مربوطه را سفارشی سازی نمایید.

ثابت‌ها	توضیحات در مورد وظایف و نقش‌های کنترل‌ها
<a href="#">Dialog.Ok</a>	یک دکمه تعریف شده با نقش پذیرنده از طرف <a href="#">AcceptRole</a>

یک دکمه تعریف شده با نقش باز کننده از طرف <code>AcceptRole</code>	<code>Dialog.Open</code>
یک دکمه تعریف شده با نقش ذخیره کننده از طرف <code>AcceptRole</code>	<code>Dialog.Save</code>
یک دکمه تعریف شده با نقش انصراف دهنده از طرف <code>RejectRole</code>	<code>Dialog.Cancel</code>
یک دکمه تعریف شده با نقش بستن از طرف <code>RejectRole</code>	<code>Dialog.Close</code>
یک دکمه از پیش تعریف شده با نقش "طرد کننده" یا "انصراف دهنده" بسته به نوع پلتفرم از طرف <code>DestructiveRole</code>	<code>Dialog.Discard</code>
یک دکمه تعریف شده با نقش اعمال کننده از طرف <code>ApplyRole</code>	<code>Dialog.Apply</code>
یک دکمه تعریف شده با نقش تنظیم کننده مجدد از طرف <code>ResetRole</code>	<code>Dialog.Reset</code>
یک دکمه تعریف شده با نقش تنظیم کننده به حالت پیش فرض مجدد از طرف <code>ResetRole</code>	<code>Dialog.RestoreDefaults</code>
یک دکمه تعریف شده با نقش کمک کننده از طرف <code>HelpRole</code>	<code>Dialog.Help</code>
یک دکمه تعریف شده با نقش ذخیره کننده کلی از طرف <code>SaveRole</code>	<code>Dialog.WriteAll</code>
یک دکمه تعریف شده با نقش پاسخ دهی مثبت "بلی" از طرف <code>YesRole</code>	<code>Dialog.Yes</code>
یک دکمه تعریف شده با نقش پاسخ دهی مثبت "بلی" سراسری از طرف <code>YesRole</code>	<code>Dialog.YesToAll</code>
یک دکمه تعریف شده با نقش پاسخ دهی منفی "خیر" از طرف <code>NoRole</code>	<code>Dialog.No</code>
یک دکمه تعریف شده با نقش پاسخ دهی منفی "خیر" سراسری از طرف <code>NoRole</code>	<code>Dialog.NoToAll</code>
یک دکمه تعریف شده با نقش سقط کننده از طرف <code>RejectRole</code>	<code>Dialog.Abort</code>
یک دکمه تعریف شده با نقش سعی کننده مجدد از طرف <code>AcceptRole</code>	<code>Dialog.Retry</code>
یک دکمه تعریف شده با نقش رد کننده "چشم پوشی کننده" از طرف <code>AcceptRole</code>	<code>Dialog.Ignore</code>
یک دکمه نامعتبر.	<code>Dialog.NoButton</code>

نکته: معمولاً این کنترل زمانی مورد استفاده قرار می‌گیرد که سوالی از کاربر پرسیده می‌شود. در این موقع ممکن است نیاز باشد تا کاربر را مجبور برای فشردن یکی از دکمه‌های ارائه شده توسط کنترل نمایید. طبیعتاً تا زمانی که وضعیت کنترل مذکور را مشخص نکرده باشد اجازه دسترسی به زیر پنجره‌ها و والد فراهم نمی‌شود. در این حالت خاصیتی با عنوان `modality` موجود است که با اختصاص دادن مقدار `Qt.WindowModal` وضعیت این خاصیت را مشخص می‌کنیم.

```
Dialog {
    id: dialog
    modality: Qt.WindowModal
    standardButtons: Dialog.Ok
}
```

## ColorDialog کنترل

کنترل `ColorDialog` امکان این را فراهم می‌سازد تا کاربر بتواند رنگ مورد نظر خود را انتخاب کند، این کنترل به طور پیشفرض نامرئی (مخفي) است. برای این کار ابتدا لازم خواهد بود تا ویژگی `visible` آن را برابر با `true` قرار دهید تا نمایان شود.

نوع	صفت
<code>color</code>	<code>color</code>
<code>color</code>	<code>currentColor</code>
<code>item</code>	<code>modality</code>
<code>bool</code>	<code>showAlphaChannel</code>
<code>string</code>	<code>title</code>

bool

visible

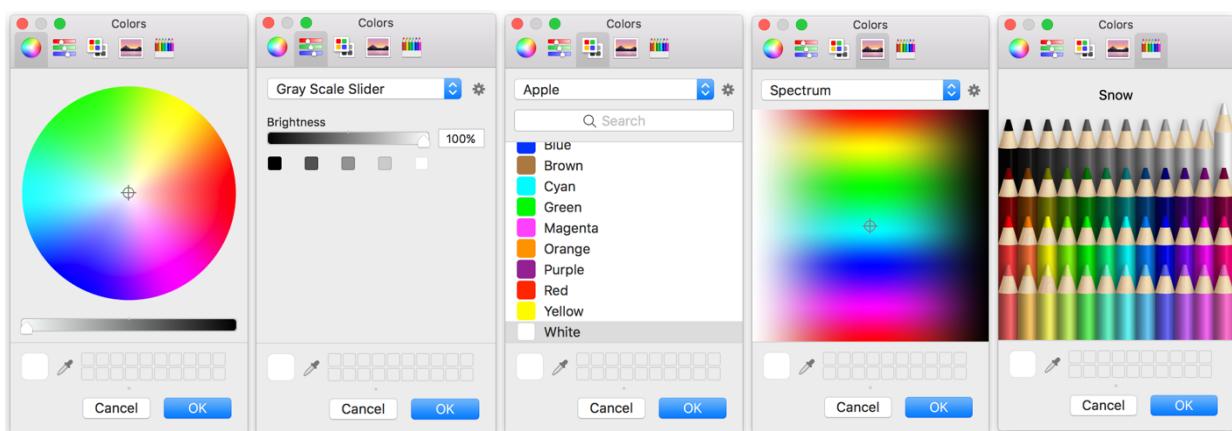
کد زیر نمونه‌ای از نحوه استفاده از این کنترل را نمایش می‌دهد:

```
ColorDialog {
    id: colorDialog
    title: "Please choose a color"

    onAccepted: {
        console.log("You chose: " + colorDialog.color)
        Qt.quit()
    }
    onRejected: {
        console.log("Canceled")
        Qt.quit()
    }

    Component.onCompleted: visible = true
}
```

با توجه به نوع سیستم‌عامل مربوطه دیالوگ (پنجره) مربوط به پلت رنگ برای کاربر در حالت‌های مختلف طبق تصاویر زیر نمایان خواهد شد.



## معرفی کنترل FileDialog

کنترل [FileDialog](#) امکان این را فراهم می‌کند تا کاربر بتواند فایل مورد نظر خود را انتخاب کند، این کنترل به طور پیشفرض نامرئی (مخفي) است.

برای این کار ابتدا لازم خواهد بود تا ویژگی `visible` آن را برابر با `true` قرار دهید تا نمایان شود.

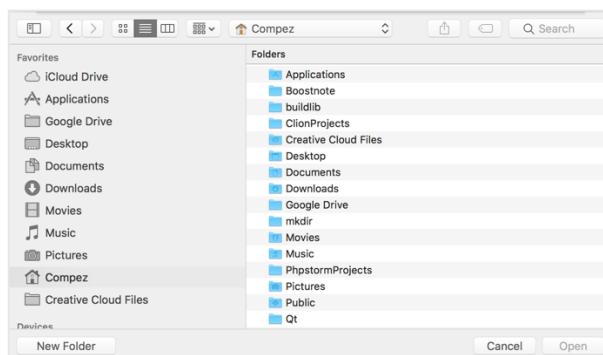
نوع	صفت
url	<code>fileUrl</code>
list<url>	<code>fileUrls</code>
url	<code>folder</code>
Qt::WindowModality	<code>modality</code>
list<string>	<code>nameFilters</code>
bool	<code>selectExisting</code>
bool	<code>selectFolder</code>
bool	<code>selectMultiple</code>
string	<code>selectNameFilter</code>

Object	<b>shortcuts</b>
bool	<b>sidebarVisible</b>
string	<b>title</b>
bool	<b>visible</b>

کد زیر نمونه‌ای از نحوه استفاده از این کنترل را نمایش می‌دهد:

```
FileDialog {
    id: fileDialog
    title: "Please choose a file"
    folder: shortcuts.home
    onAccepted: {
        console.log("You chose: " + fileDialog.fileUrls)
        Qt.quit()
    }
    onRejected: {
        console.log("Canceled")
        Qt.quit()
    }
    Component.onCompleted: visible = true
}
```

نتیجه کد فوق به صورت زیر خواهد بود که امکان انتخاب فایل، سوئیچ بین دایرکتوری‌ها و حتی ساختن مسیری جدید را فراهم می‌کند.



## FontDialog کنترل

کنترل [FontDialog](#) امکان این را فراهم می‌کند تا کاربر بتواند قلم (فونت) مورد نظر خود را انتخاب کند. این کنترل به طور پیشفرض نامرئی (مخفي) است. برای این کار ابتدا لازم خواهد بود تا ویزگی `visible` آن را برابر با `true` قرار دهید تا نمایان شود.

نوع	صفت
QPushButton	<b>currentFont</b>
QObject	<b>font</b>
item	<b>modality</b>
bool	<b>showAlphaChannel</b>
string	<b>title</b>
bool	<b>visible</b>

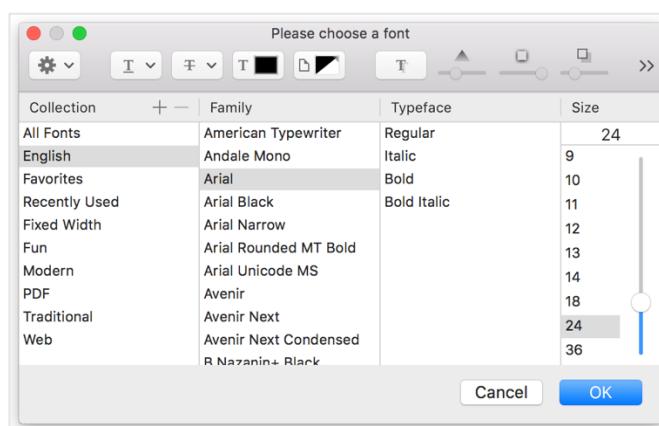
کد زیر نمونه‌ای از نحوه استفاده از این کنترل را نمایش می‌دهد:

```

FontDialog {
    id: fontDialog
    title: "Please choose a font"
    font: Qt.font({ family: "Arial", pointSize: 24, weight: Font.Normal })
    onAccepted: {
        console.log("You chose: " + fontDialog.font)
        Qt.quit()
    }
    onRejected: {
        console.log("Canceled")
        Qt.quit()
    }
    Component.onCompleted: visible = true
}

```

با توجه به نوع سیستم عامل مربوطه دیالوگ (پنجره) مربوط به جعبه قلم برای کاربر در حالت‌های مختلف طبق تصاویر زیر نمایان خواهد شد.



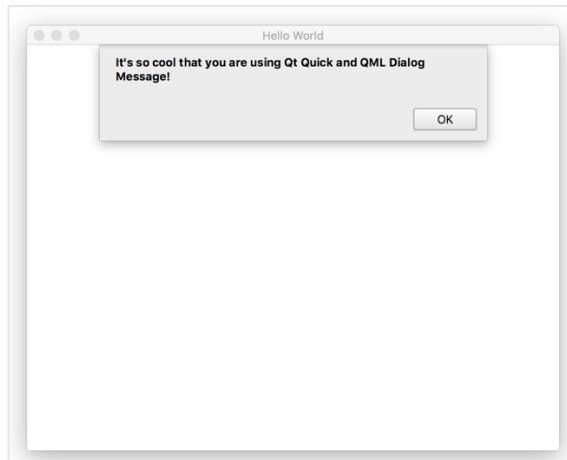
## معرفی کنترل **MessageDialog**

کنترل **MessageDialog** یکی از اساسی‌ترین کنترل‌هایی که مورد استفاده قرار می‌گیرد کنترل هشدار است. این کنترل به طور پیشفرض نامرئی (مخفي) است. برای این کار ابتدا لازم خواهد بود تا ویژگی `visible` آن را برابر با `true` قرار دهید تا نمایان شود.

نوع	صفت
QPushButton	<code>clickedButton</code>
string	<code>detailedText</code>
QQuickStandardIcon::Icon	<code>icon</code>
string	<code>informativeText</code>
Qt::WindowModality	<code>modality</code>
StandardButtons	<code>standardButtons</code>
string	<code>text</code>
string	<code>title</code>
bool	<code>visible</code>

کد زیر نمونه‌ای از نحوه استفاده از این کنترل را نمایش می‌دهد:

```
MessageDialog {
    id: messageDialog
    title: "Hello, I am C++ with QML"
    text: "It's so cool that you are using Qt Quick and QML Dialog Message!"
    onAccepted: {
        console.log("And of course you could only agree.")
        Qt.quit()
    }
    Component.onCompleted: visible = true
}
```



ممکن است نیاز باشد از دکمه‌های بیشتری بر روی پیغام خود استفاده کنید، در این صورت کافی است با مقدار دهی به خاصیت standardButtons دکمه‌های مورد نیاز خود را وارد کنید.

## Layout معرفی

نوع Layout خصوصیات درج آیتم بر روی لایه‌های ColumnLayout، GridLayout و RowLayout را فراهم می‌کند. برای مثال هریک از انواع ذکر شده می‌توانند دارای خصوصیاتی باشند که در Layout به آنها اشاره می‌شود. برای دسترسی به این موارد کافی است ماثول آن را صدا بزنید:

```
import QtQuick.Layouts 1.3
```

صفت	نوع
alignment	Qt.Alignment
bottomMargin	real
column	int
columnSpan	int
fillHeight	bool

<b>fillWidth</b>	bool
<b>leftMargin</b>	real
<b>margins</b>	real
<b>maximumHeight</b>	real
<b>maximumWidth</b>	real
<b>minimumHeight</b>	real
<b>minimumWidth</b>	real
<b>preferredHeight</b>	real
<b>preferredWidth</b>	real
<b>rightMargin</b>	real
<b>row</b>	int
<b>rowSpan</b>	int
<b>topMargin</b>	real

خاصیت‌های موجود در Layout در سرتاسر یک پروژه تحت فناوری Qt Quick بسیار مهم هستند. خاصیت‌های فوق در ادامه تعریف و نحوه استفاده از آن‌ها آورده شده است.

مشخصه alignment این امکان را فراهم می‌سازد تا شما آیتمی را بر اساس ترازی که نیاز است به نمایش در آورید. برای استفاده از آن کافی است کنترل، لایه و یا اشیاء خود را به صورت زیر داشته باشید.

توضیحات جهت همتراز سازی در حالت افقی	ارزش	ثابت‌ها
ترازبندی از سمت (لبه) چپ.	0x0001	Qt::AlignLeft
ترازبندی از سمت (لبه) راست.	0x0002	Qt::AlignRight
در فضای موجود به صورت افقی ترازبندی می‌شود.	0x0004	Qt::AlignHCenter
همتراز کردن متن در فضای موجود.	0x0008	Qt::AlignJustify

توضیحات جهت همتراز سازی در حالت عمودی	ارزش	ثابت‌ها
ترازبندی از سمت (لبه) بالا.	0x0020	Qt::AlignTop
ترازبندی از سمت (لبه) پایین.	0x0040	Qt::AlignBottom
در فضای موجود به صورت عمودی ترازبندی می‌شود.	0x0080	Qt::AlignVCenter
ترازبندی در راستای پایه.	0x0100	Qt::AlignBaseline

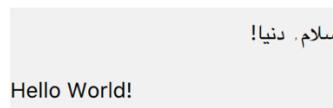
برای معرفی نحوه عملکرد Layout قبل از هرچیز باید توجه داشته باشید که تنها با استفاده از ColumnLayout، GridLayout و RowLayout می‌توانید از خاصیت‌های آن بر روی آیتم اعمال کنید چرا که آیتم یا شیء مورد نظر توسط آن‌ها نگهداری می‌شوند. در زیر مثالی آورده شده است که متن "سلام، دنیا!" در سمت (لبه) راست و متن لاتین "Hello, World!" در سمت (لبه) چپ لایه ColumnLayout ترازبندی شده است.

```
Rectangle {
    color: "#f1f1f1"
    width: 200
    height: 64
```

```

anchors.centerIn: parent
ColumnLayout {
    anchors.fill: parent
    Text {
        text: qsTr("سلام، دنیا!")
        Layout.alignment: Qt.AlignRight
    }
    Text {
        text: qsTr("Hello World!")
        Layout.alignment: Qt.AlignLeft
    }
}

```



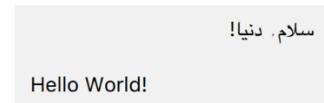
همانطور که مشاهده می‌کنید متن‌های مورد نظر در سمت راست و چپ شیء Rectangle با استفاده از ColumnLayout و ویژگی‌های Layout ترازبندی شده‌اند.

برای استفاده از دیگر ویژگی‌های Layout به همین ترتیب باید عمل کنیم، لذا چند نمونه مثال دیگری را در ادامه آورده‌ایم. فرض کنید نیاز است حاشیه‌های مثال بالا را که متن‌های مربوطه به لبه‌های شیء Rectangle چسبیده اند فاصله دار کنیم. در این صورت کافی است از خاصیت leftMargin و rightMargin در آن استفاده شود.

```

Text {
    text: qsTr("سلام، دنیا!")
    Layout.alignment: Qt.AlignRight
    Layout.rightMargin: 10
}
Text {
    text: qsTr("Hello World!")
    Layout.alignment: Qt.AlignLeft
    Layout.leftMargin: 10
}

```



نتیجه کد فوق به صورت تصویر بالا خواهد بود که هر دو متن از سمت مرتبط با خود با مقدار ۱۰ پیکسل فاصله پیدا کرده اند. فرض کنید نیاز باشد حاشیه بندی به صورت کلی در داخل خود ColumnLayout صورت بگیرد در این صورت نیاز خواهد داشت تا از خاصیت anchors (لنگر) اندازی استفاده کنید. نمونه مثال کد آن به صورت زیر خواهد بود:

```

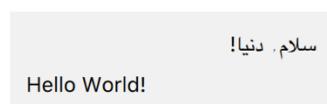
Rectangle {
    color: "#f1f1f1"
    width: 200
    height: 64
    anchors.centerIn: parent
    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 10
        Text {

```

```

        text: qsTr("سلام، دنیا!")
        Layout.alignment: Qt.AlignRight
    }
    Text {
        text: qsTr("Hello World!")
        Layout.alignment: Qt.AlignLeft
    }
}

```



در این روش هر آیتمی به جز دو آیتم متن در داخل لایه ColumnLayout قرار بگیرد با فاصله حاشیه از تمامی جهات (بالا، پایین، راست و چپ) نمایش داده خواهد شد. در واقع حاشیه بندی از تمامی جهات صورت می‌گیرد و نیازی نخواهد بود برای هریک از آیتم‌های موجود در داخل لایه Layout خاصیت‌های استفاده شود.

در ادامه راجح به خاصیت‌های دیگر و نحوه سفارشی سازی فرم و آیتم‌ها در لایه‌های نگه دارنده مثال‌های آورده شده است که نشان می‌دهد چگونه می‌توان یک فرم همراه با آیتم‌های مورد نظر را به صورت سفارشی ترازبندی کرد.

## معرفی ColumnLayout

نوع ColumnLayout خصوصیات درج آیتم بر روی لایه به شیوه ستونی را فراهم می‌کند. در این لایه می‌توانیم آیتم یا اشیاء مورد نظر را به ترتیب ستونی درج نموده و هریک از آن‌ها را بر اساس نیاز ترازبندی کنیم.

صفت	نوع
<b>layoutDirection</b>	enumeration
<b>space</b>	real

در زیر کدی آورده شده است که به ترتیب آیتم‌هایی را بر اساس اندازه‌های ثابت شده در قالب ستون نمایش می‌دهد:

```

ColumnLayout {
    spacing: 2

    Rectangle {
        Layout.alignment: Qt.AlignCenter
        color: "#e200ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 20
    }

    Rectangle {
        Layout.alignment: Qt.AlignRight
        color: "#bb00ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 40
    }

    Rectangle {
        Layout.alignment: Qt.AlignBottom
        Layout.fillHeight: true
    }
}

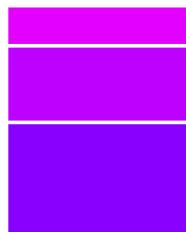
```

```

        color: "#8800ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 60
    }
}

```

همانطور که در کد فوق مشاهده می‌کنید لایه ColumnLayout شامل ۳ عدد شیء از نوع Rectangle است که هریک از آن‌ها دارای ویژگی‌های preferredHeight و preferredWidth می‌باشند که اجازه می‌هد تا ارتفاع و طول آن‌ها به صورت ترجیحی تعیین شوند. توجه داشته باشید که صفت spacing در ColumnLayout عمل فاصله زدن بین آیتم‌های داخل خودش را دارد که به صورت زیر خواهد بود.



در ادامه لازم است به یکی از مهمترین و پرکاربردترین صفت‌های لایه اشاره کنیم که برای برنامه‌های فارسی که از ترازبندی راست به چپ استفاده می‌کنند مطل乎 شویم که چطور می‌توان توسط این صفت آیتم مورد نظر را در قالب راست چین یا چپ چین طراحی کرد. در حالت عادی کد زیر را خواهیم داشت که در ادامه نتیجه خروجی آن را به صورت چپ به راست مشاهده می‌کنیم.

```

ColumnLayout {
    spacing: 2
    anchors.centerIn: parent

    Rectangle {
        Layout.alignment: Qt.AlignRight
        color: "orange"
        Layout.preferredWidth: 200
        Layout.preferredHeight: 40

        RowLayout {
            anchors.centerIn: parent
            Text {text: "نام و نام خانوادگی";
                  color: "#fff";
                  font.bold: true;
            }
            Text {text: "کامبیز اسدزاده";
                  color: "#fff";
            }
        }
    }
}

```

کامبیز اسدزاده نام و نام خانوادگی:

قرار است آیتم‌های عنوان و پاسخ به ترتیب راست به چپ نمایش داده شوند:

```
RowLayout {
```

```

layoutDirection: Qt.RightToLeft
anchors.centerIn: parent
Text {text: "نام و نام خانوادگی";
      color: "#fff";
      font.bold: true
    }
Text {text: "کامبیز اسدزاده";
      color: "#fff";
    }
}

```

با درج کد layoutDirection و مقدار Qt.RightToLeft دستور این کهایتم‌ها به صورت راست به چپ نمایش داده شوند به نوع RowLayout صادر می‌شود.

نام و نام خانوادگی: کامبیز اسدزاده

مثال زیر از مثال یک فرم اطلاعاتی بسیار ساده است که با استفاده از Layout و layoutDirection از چین نمایش داده می‌شوند.

```

ColumnLayout{
  spacing: 2
  anchors.centerIn: parent

  Rectangle {
    color: "gray"
    Layout.preferredWidth: 200
    Layout.preferredHeight: 40

    RowLayout {
      layoutDirection: Qt.RightToLeft
      anchors.centerIn: parent
      Text {text: "نام و نام خانوادگی";
            color: "#fff";
            font.bold: true;
          }
      Text {text: "کامبیز اسدزاده";
            color: "#fff";
          }
    }
  }

  Rectangle {
    color: "gray"
    Layout.preferredWidth: 200
    Layout.preferredHeight: 40

    RowLayout {
      layoutDirection: Qt.RightToLeft
      anchors.centerIn: parent
      Text {text: "سن:";
            color: "#fff";
            font.bold: true;
          }
      Text {text: "۲۸";
            color: "#fff";
          }
    }
  }
}

```

```
        }

    Rectangle {
        color: "gray"
        Layout.preferredWidth: 200
        Layout.preferredHeight: 40

        RowLayout {
            layoutDirection: Qt.RightToLeft
            anchors.centerIn: parent
            Text {text: "شركـت:";
                  color: "#fff";
                  font.bold: true;
            }
            Text {text: "دـات ويـوز";
                  color: "#fff";
            }
        }
    }

    Rectangle {
        color: "gray"
        Layout.preferredWidth: 200
        Layout.preferredHeight: 40

        RowLayout {
            layoutDirection: Qt.RightToLeft
            anchors.centerIn: parent
            Text {text: "عنـوان:";
                  color: "#fff";
                  font.bold: true;
            }
            Text {text: "مدـير عـامل";
                  color: "#fff";
            }
        }
    }

}
```



## GridLayout معرفی

یکی از مهمترین انواع لایه‌ها لایه GridLayout است که به صورت یک توری عمل می‌کند و در صورتی که فرم تغییر اندازه دهد آیتم‌های موجود بر روی آن نیز بر اساس محیط تغییر اندازه خواهند داد.

صفت	نوع
<b>columnSpacing</b>	real
<b>columns</b>	int
<b>flow</b>	enumeration
<b>layoutDirection</b>	enumeration
<b>rowSpacing</b>	real
<b>rows</b>	int

کد زیر نحوه عملکرد آن را نشان می‌دهد که با تعیین مشخصه‌های Layout.fillwidth و Layout.fillHeight امکان هماهنگی در ابعاد آیتم موقع تغییر اندازه فرم از طریق GridLayout فراهم می‌شود.

```
GridLayout {
    id: gridLayout
    anchors.fill: parent
    Rectangle {
        Layout.row: 0
        Layout.column: 0
        Layout.fillHeight: true
        Layout.fillWidth: true
        color: "green"
    }
    Rectangle {
        Layout.row: 0
        Layout.column: 1
        Layout.fillHeight: true
        Layout.fillWidth: true
        color: "orange"
    }
}
```

کد زیر نمونه‌ای است از لایه Grid همراه با آیتم‌های مختلف که نشان می‌دهد چگونه می‌توان آیتم‌ها را بروی لایه ترازبندی و یکنواخت ساخت.

```
import QtQuick 2.9
import QtQuick.Window 2.2
import QtQuick.Dialogs 1.2
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.3

Window {
    visible: true
    title: qsTr("Hello World")

    property int margin: 11
    width: 600
    height: 500
    minimumWidth: mainLayout.Layout.minimumWidth + 2 * margin
    minimumHeight: mainLayout.Layout.minimumHeight + 2 * margin

    ColumnLayout {
        id: mainLayout
        anchors.fill: parent
    }
}
```

```

anchors.margins: margin
GroupBox {
    id: rowBox
    title: "Enter your message"
    Layout.fillWidth: true

    RowLayout {
        id: rowLayout
        anchors.fill: parent
        TextField {
            placeholderText: "I like C++ programming..."
            Layout.fillWidth: true
        }
        Button {
            text: "Send"
        }
    }
}

GroupBox {
    id: gridBox
    title: "Grid layout"
    Layout.fillWidth: true

    GridLayout {
        id: gridLayout
        rows: 3
        flow: GridLayout.TopToBottom
        anchors.fill: parent

        Label { text: "Line 1" }
        Label { text: "Line 2" }
        Label { text: "Line 3" }

        TextField { }
        TextField { }
        TextField { }

        TextArea {
            text: "C++ (pronounced cee plus plus /'si: plʌs plʌs/) is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation."
            Layout.rowSpan: 3
            Layout.fillHeight: true
            Layout.fillWidth: true
            wrapMode: TextArea.Wrap
        }
    }
}
TextArea {
    id: t3
    text: "C++ (pronounced cee plus plus /'si: plʌs plʌs/) is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation."
    Layout.minimumHeight: 30
    Layout.fillHeight: true
    Layout.fillWidth: true
    wrapMode: TextArea.Wrap
}

```

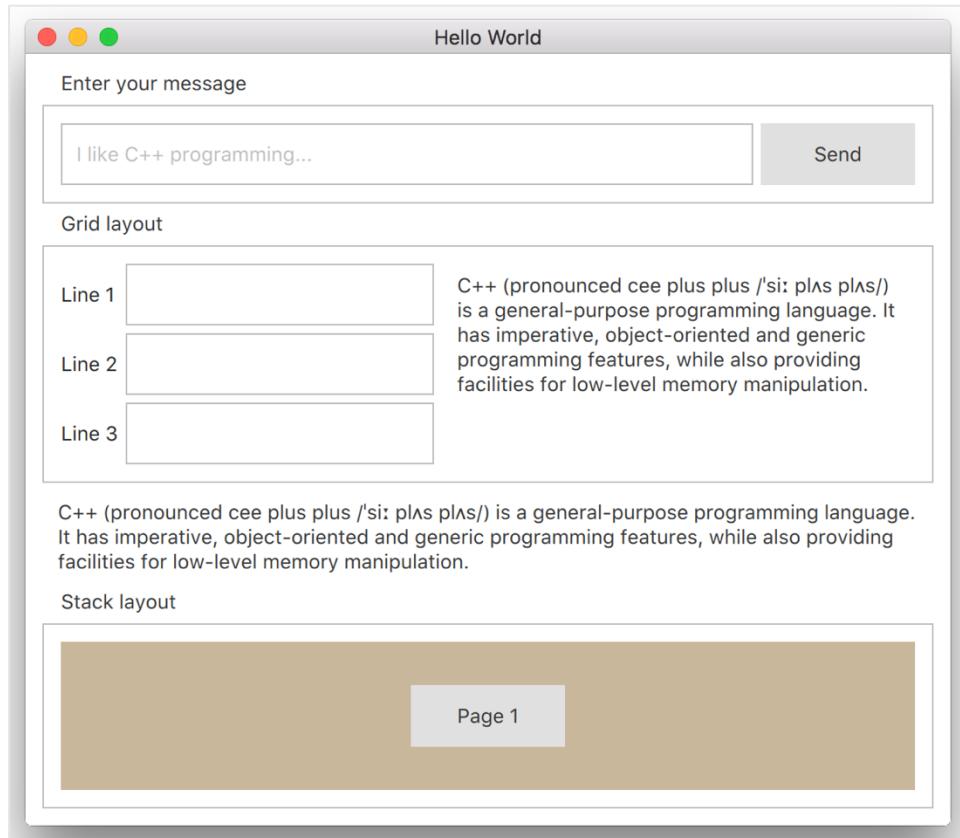
```

GroupBox {
    id: stackBox
    title: "Stack layout"
    implicitWidth: 200
    implicitHeight: 160
    Layout.fillWidth: true
    Layout.fillHeight: true
    StackLayout {
        id: stackLayout
        anchors.fill: parent

        function advance() { currentIndex = (currentIndex + 1) % count }

        Repeater {
            id: stackRepeater
            model: 5
            Rectangle {
                color: Qt.hsla((0.5 + index)/stackRepeater.count, 0.3,
0.7, 1)
                Button { anchors.centerIn: parent; text: "Page " + (index
+ 1); onClicked: { stackLayout.advance() } }
            }
        }
    }
}

```



نمونه فوق به گونه‌ای پیاده سازی شده است که تمامی آیتم‌های موجود بر روی آن موقع تغییر اندازه فرم یا والد اصلی واکنشگرا می‌شوند.

نوع RowLayout خصوصیات درج آیتم بر روی لایه به شیوه ردیفی را فراهم می‌کند. در این لایه می‌توانیم آیتم یا اشیاء مورد نظر را به ترتیب ردیف درج نموده و هریک از آن‌ها را بر اساس نیاز ترازبندی کنیم.

صفت	نوع
<b>layoutDirection</b>	enumeration
<b>space</b>	real

در زیر کدی آورده شده است که به ترتیب آیتم‌هایی را بر اساس اندازه‌های ثابت شده در قالب ستون نمایش می‌دهد:

```
RowLayout {
    spacing: 2

    Rectangle {
        Layout.alignment: Qt.AlignCenter
        color: "#e200ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 20
    }

    Rectangle {
        Layout.alignment: Qt.AlignRight
        color: "#bb00ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 40
    }

    Rectangle {
        Layout.alignment: Qt.AlignBottom
        Layout.fillHeight: true
        color: "#8800ff"
        Layout.preferredWidth: 100
        Layout.preferredHeight: 60
    }
}
```

همانطور که در کد فوق مشاهده می‌کنید لایه RowLayout شامل ۳ عدد شیء از نوع Rectangle است که هریک از آن‌ها دارای ویژگی‌های Layout مخصوص خود با صفت preferredWidth و preferredHeight می‌باشد که اجازه می‌هد تا ارتفاع و طول آن‌ها به صورت ترجیحی تعیین شوند. توجه داشته باشید که صفت spacing در ColumnLayout همانند نوع قابلی عمل فاصله زدن بین آیتم‌های داخل خودش را دارد که به صورت زیر خواهد بود.



نوع همانطور که از نامش مشخص است امکان این را فراهم می‌کند که آیتم‌های موجود را به صورت پشته در اختیار نگه داشته و توسط مشخصه `currentIndex` آنها را به نمایش در آورد.

صفت	نوع
<code>count</code>	<code>int</code>
<code>currentIndex</code>	<code>int</code>

کد زیر نمونه‌ای از نحوه استفاده از `StackLayout` را نشان می‌دهد که هر یک از آیتم‌ها در پشته لایه قرار گرفته و توسط شمارنده شاخص یعنی `currentIndex` مقدار آیتم مرتبط با خود را نمایش می‌دهد. در کد زیر شاخص برابر است با ۱ بدین معنی است که شیء از نوع `Rectangle` با رنگ `Two` نارنجی و متن `Two` نمایش داده خواهد شد.

### `StackLayout` {

```

id: layout
anchors.fill: parent
currentIndex: 1

Rectangle {
    color: 'green'
    implicitWidth: 100
    implicitHeight: 100
    Text {
        text: qsTr("One")
        anchors.centerIn: parent
        color: "#fff"
    }
}
Rectangle {
    color: 'orange'
    implicitWidth: 100
    implicitHeight: 100
    Text {
        text: qsTr("Two")
        anchors.centerIn: parent
        color: "#fff"
    }
}
Rectangle {
    color: 'blue'
    implicitWidth: 100
    implicitHeight: 100
    Text {
        text: qsTr("Three")
        anchors.centerIn: parent
        color: "#fff"
    }
}
}
```

کاربرد این لایه در جمع آوری و نمایش لایه‌ها بسیار مهم است، همچنین می‌توان با گرفتن مقدار `count` که وظیفه دریافت شاخص مربوط با آیتم‌های موجود در لایه را دارد شرایط موجود را بررسی کرد.

در ادامه مثالی آورده شده است که با دریافت شماره شاخص (`index`) از طرف کاربر آیتم مربوط به آن شاخص را نمایش خواهد داد:

```

ColumnLayout {
    anchors {
        margins: 20
        fill: parent
    }

    RowLayout {
        spacing: 10
        Label {text: qsTr("Enter your page index number:")}
        TextField {
            id:indexInput
        }
        Button {
            id:showIndexButton
            text: "Select Index";
            onClicked: {
                layout.currentIndex = indexInput.text
            }
        }
    }

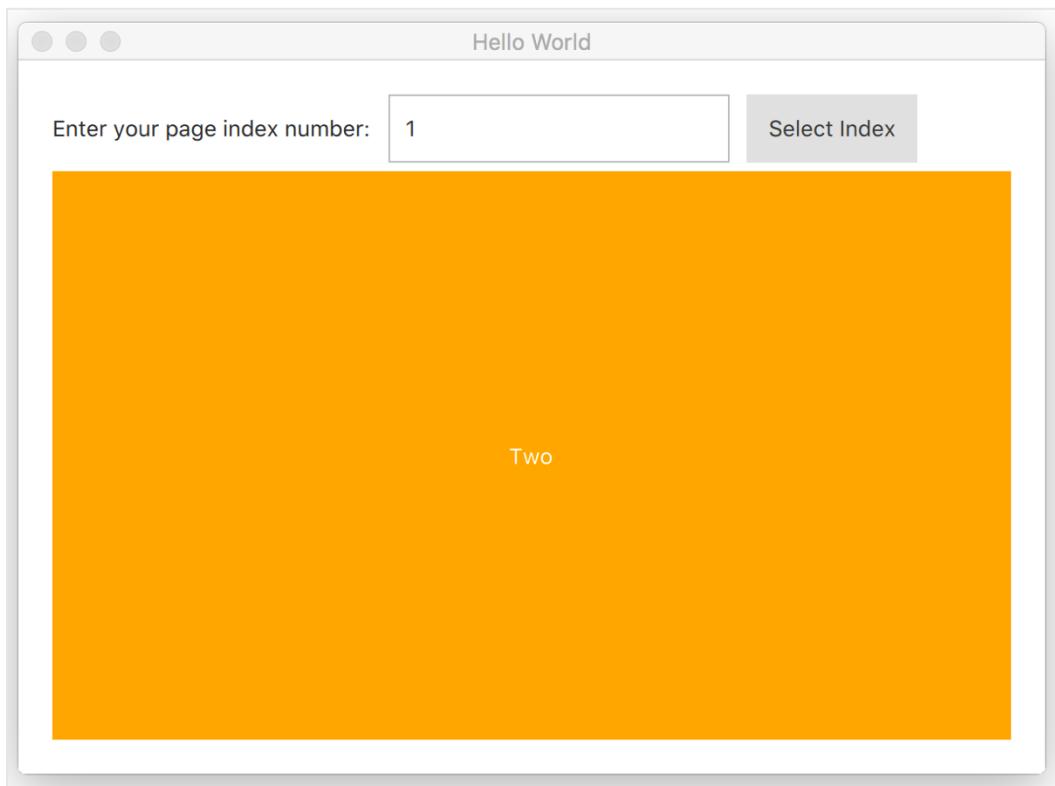
    StackLayout {
        id: layout
        currentIndex: 1

        Rectangle {
            color: 'green'
            implicitWidth: 100
            implicitHeight: 100
            Text {
                text: qsTr("One")
                anchors.centerIn: parent
                color: "#fff"
            }
        }
        Rectangle {
            color: 'orange'
            implicitWidth: 100
            implicitHeight: 100
            Text {
                text: qsTr("Two")
                anchors.centerIn: parent
                color: "#fff"
            }
        }
        Rectangle {
            color: 'blue'
            implicitWidth: 100
            implicitHeight: 100
            Text {
                text: qsTr("Three")
                anchors.centerIn: parent
                color: "#fff"
            }
        }
    }
}

```

در خروجی زیر با انتخاب شماره شاخص (index) آیتم مربوط به آن نشان داده خواهد شد.

نکته: توجه داشته باشید که در برنامه‌نویسی شاخص همیشه از ۰ آغاز می‌شود. بنابراین مقدار ۱ برابر خواهد بود با شاخص آیتم دوم.



## معرفی Qt Quick Control Style

با توجه به مباحث تجربه‌کاربری (UX) و رابط‌کاربری (UI) که مهمترین بخشی از یک محصول بشمار می‌آیند طراحی و پوسته نویسی برای محصول در برنامه‌های تحت C++ توسط QML با ترکیبی از فناوری‌های HTML و JavaScript, CSS و CSS با ترکیبی از فناوری‌های HTML و JavaScript, CSS و CSS در کنار یکدیگر لازمه داشتن دانش طراحی در بخش فرانت‌اند است که البته برای برنامه نویسان فول‌استک این امکان بسیار جذاب است که خود بتواند در سریعترین روش ممکن ترکیب کدهای بک‌اند ای خود را با فرانت‌اند ایجاد کند.

با توجه به اهداف پشتیبانی از GUI در کتابخانه کیوت این بخش از طراحی محصول به خوبی پشتیبانی شده است و کاربر می‌تواند هر آنچه که در ذهن خلاق خود دارد پیاده سازی نماید.

ماژول Qt Quick Control Style این را فراهم می‌کند تا بتوانید پوسته مورد نظر خود را برای کنترل‌های موجود باز طراحی و سفارشی کنید. البته به این نکته توجه داشته باشید که استایل نویسی در سرتاسر محصول ساخته شده توسط C++ و QML ممکن است و در صورتی که شما کنترلی را به صورت یک ماژول سفارشی طراحی کرده‌اید که بر پایه Qt Quick Control است. در این صورت کافی است طراحی مجدد استایل از سمت C++ و یا QML پوسته مورد نظرتان را برای کنترل مورد نظر خود طراحی کنید.

قبل از آغاز طراحی پوسته کتابخانه کیوت خود پوسته‌های پیشفرض استانداردی را برای طراحان فراهم آورده است که بر پایه سه طرح استاندارد Default و Universal و Material ارائه داده است. که به ترتیب بر پایه استانداردهای Microsoft, Google و پیش فرض Qt Quick Style است.

برای دسترسی به این مکان کافی است فایل **qtquickcontrols2.conf** را در پروژه خود اضافه کنید. سپس کد مربوط به آن فایل به صورت زیر خواهد بود که در نسخه ۵.۷ به بعد کیوت ارائه شده است.

```
; This file can be edited to change the style of the application
; See Styling Qt Quick Controls 2 in the documentation for details:
; http://doc.qt.io/qt-5/qtquickcontrols2-styles.html

[Controls]
Style=Default

[Universal]
Theme=Light
;Accent=Steel

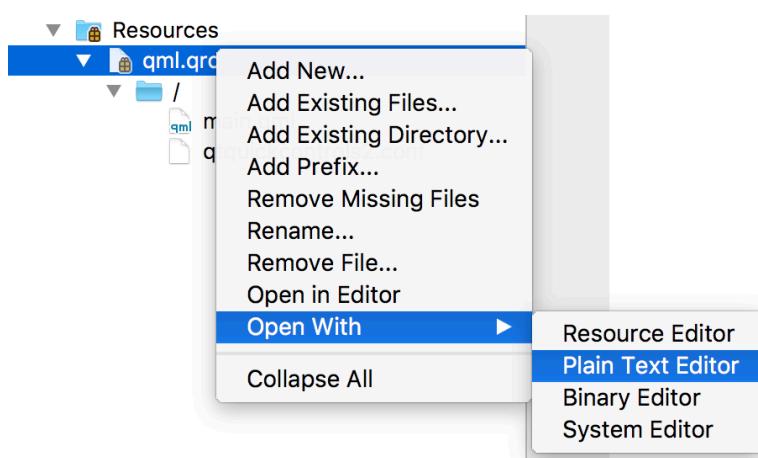
[Material]
Theme=Light
;Accent=BlueGrey
;Primary=BlueGray
```

در کد فوق مقدار [Controls] با متغیر Style ارزیابی می‌شود. کافی است عنوان Universal و یا Material را در آن قرار دهید تا ویژگی‌های ظاهری مربوط به هر یک را پروژه ارت بری کند.

کد پیکربندی فایل مربوطه در فایل qrc به صورت زیر است:

```
<RCC>
<qresource prefix="/">
    <file>main.qml</file>
    <file>qtquickcontrols2.conf</file>
</qresource>
</RCC>
```

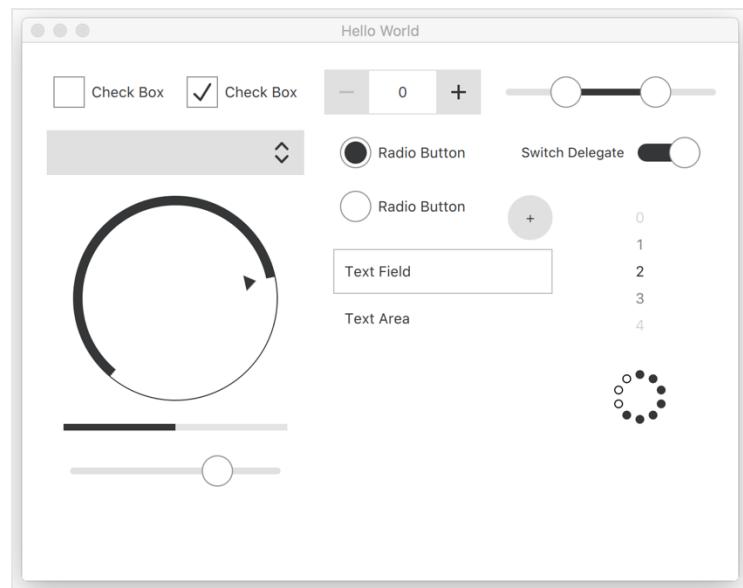
کد فوق را با راست کلیک بر روی فایل qml.qrc و انتخاب Open With Plain Text Editor ایجاد و وارد نمایید.



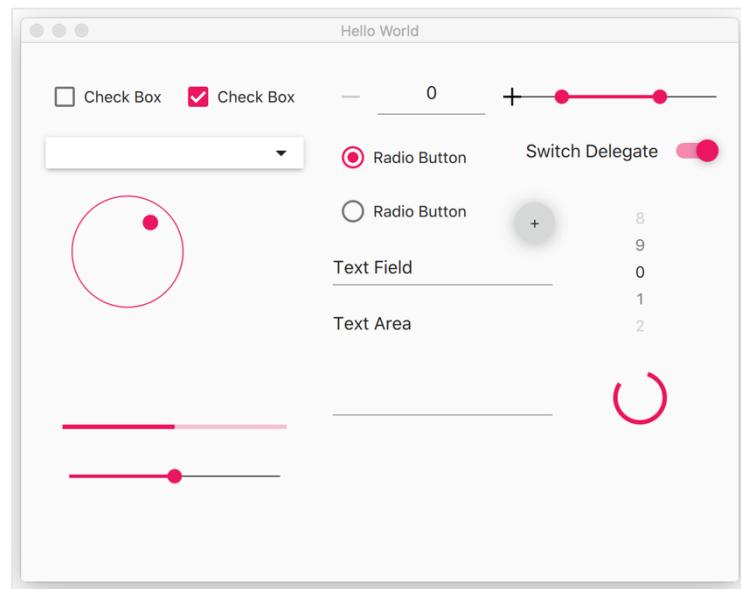
سپس کد زیر را در فایل pro. مربوط به پروژه وارد خواهیم کرد.

```
RESOURCES += qml.qrc
```

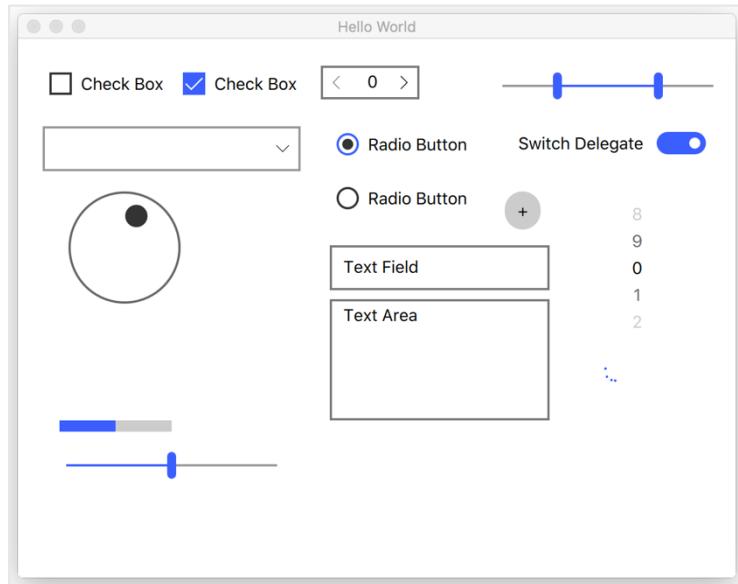
با فرض اینکه مقدار Style را برابر مقدار Default باشد طرح پوسه برنامه به صورت زیر خواهد بود:



در صورتی که مقدار Style را بر روی Material قرار دهیم برنامه به سبک گوگل متريال تنظيم خواهد شد.



همچنین طرح Universal در رابطه با سبک جدید ویندوز به صورت زیر خواهد بود.



نکته جالبی که در این بحث موجود است امکان تغییر رنگ و پوسته طرح‌های استاندارد فوق فراهم شده است. برای مثال طرح متریال گوگل را می‌توان به رنگ‌های دلخواه دیگری تغییر دهیم که در این صورت کافی است رنگ‌های مربوطه را در متغیرهای Primary و Accent اعمال کنیم که لیست آن‌ها در ادامه آمده است. فراموش نکنید جهت اعمال رنگ‌های مورد نظر سیمی‌کالون (;) را از ابتدای متغیرها حذف کنید. این کار عمل غیرفعال‌سازی استفاده از متغیرها را اعمال می‌کند.

#### [Material]

**Theme=Light** // (Dark/Light) : روشن / تیره

**Accent=BlueGrey**

**Primary=BlueGray**

در رنگ‌بندی پوسته‌های متریال کد رنگ‌های استاندارد از پیش تعریف شده‌ای وجود دارند که لیست آن به صورت زیر آمده است که مناسب برای طراحی‌های روشن است (Light Mode) که هریک از آن‌ها را می‌توانید برابر متغیرهای Primary و Accent قرار دهید.

ثابت‌ها	توضیح / کد رنگ
Material.Red	#F44336
Material.Pink	#E91E63 (قدر پیشفرض برای accent)
Material.Purple	#9C27B0
Material.DeepPurple	#673AB7
Material.Indigo	#3F51B5 (قدر پیشفرض برای primary)
Material.Blue	#2196F3
Material.LightBlue	#03A9F4
Material.Cyan	#00BCD4

Material.TeaL	#009688
Material.Green	#4CAF50
Material.LightGreen	#8BC34A
Material.Lime	#CDDC39
Material.Yellow	#FFEB3B
Material.Amber	#FFC107
Material.Orange	#FF9800
Material.DeepOrange	#FF5722
Material.Brown	#795548
Material.Grey	#9E9E9E
Material.BlueGrey	#607D8B

همچنین رنگ استایل‌های از پیش تعریف شده‌ای برای پوسته‌های تیره تعریف شده‌اند که لیست آن در ادامه آمده است:

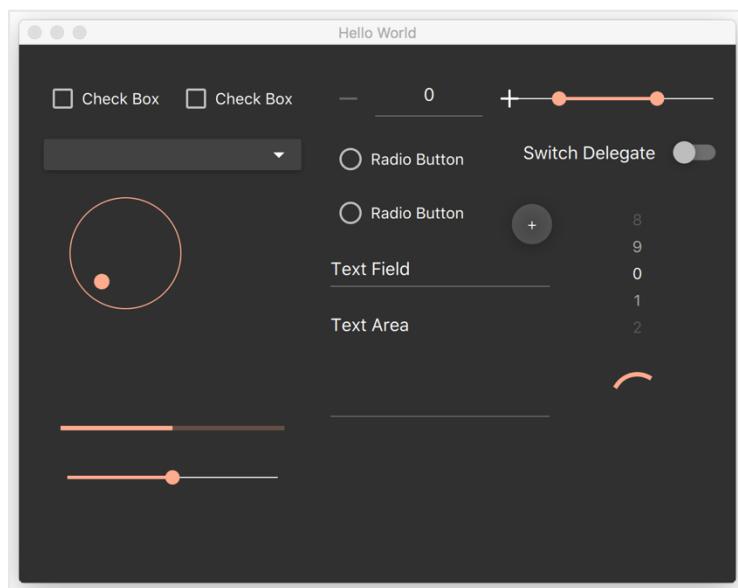
ثابت‌ها	توضیح / کد رنگ
Material.Red	#EF9A9A
Material.Pink	#F48FB1 مقدار پیشفرض برای accent)
Material.Purple	#CE93D8
Material.DeepPurple	#B39DDB
Material.Indigo	#9FA8DA مقدار پیشفرض برای primary)
Material.Blue	#90CAF9
Material.LightBlue	#81D4FA
Material.Cyan	#80DEEA
Material.TeaL	#80CBC4
Material.Green	#A5D6A7

Material.LightGreen	#C5E1A5
Material.Lime	#E6EE9C
Material.Yellow	#FFF59D
Material.Amber	#FFE082
Material.Orange	#FFCC80
Material.DeepOrange	#FFAB91
Material.Brown	#BCAAA4
Material.Grey	#EEEEEE
Material.BlueGrey	#B0BEC5

جهت استفاده از پوسته‌های تیره مقدار مربوط به Theme را باید برابر با Dark قرار دهید.

```
[Material]
Theme=Dark
Accent=BlueGrey
Primary=BlueGray
```

استفاده از استایل تیره سبکی به روش زیر را فراهم می‌آورد :



در ادامه لیست کد رنگ‌های مخصوص پوسته ویندوز (Universal) به صورت زیر است:

ثابت‌ها	توضیح / کد رنگ
Universal.Lime	#A4C400

Universal.Green	#60A917
Universal.Emerald	#008A00
Universal.Teal	#00ABA9
Universal.Cyan	#1BA1E2
Universal.Cobalt	#3E65FF (default accent)
Universal.Indigo	#6A00FF
Universal.Violet	#AA00FF
Universal.Pink	#F472D0
Universal.Magenta	#D80073
Universal.Crimson	#A20025
Universal.Red	#E51400
Universal.Orange	#FA6800
Universal.Amber	#FOA30A
Universal.Yellow	#E3C800
Universal.Brown	#825A2C
Universal.Olive	#6D8764
Universal.Steel	#647687
Universal.Mauve	#76608A
Universal.Taupe	#87794E

علاوه بر روش Qt Quick C++ تنظیم پوسته با وارد کردن ماتژول qtquickcontrols2 در فایل pro. پروژه فراهم می‌شود. در این صورت تنها کافی است با فراخوانی سرآیند زیر دسترسی به تعیین پوسته مورد نظر را فعال سازیم.  
ابتدا در فایل pro. نام ماتژول را وارد کنید:

```
QT += qml quick quickcontrols2
```

سپس کد C++ آن به صورت زیر خواهد بود که با فراخوانی فایل سرآیند با نام QQuickStyle امکان دسترسی به کلاس QQuickStyle فراهم می‌شود.

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQuickStyle>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);

    QQuickStyle::setStyle("Material");

    QQmlApplicationEngine engine;
    engine.load(QUrl(QLatin1String("qrc:/main.qml")));

    return app.exec();
}
```

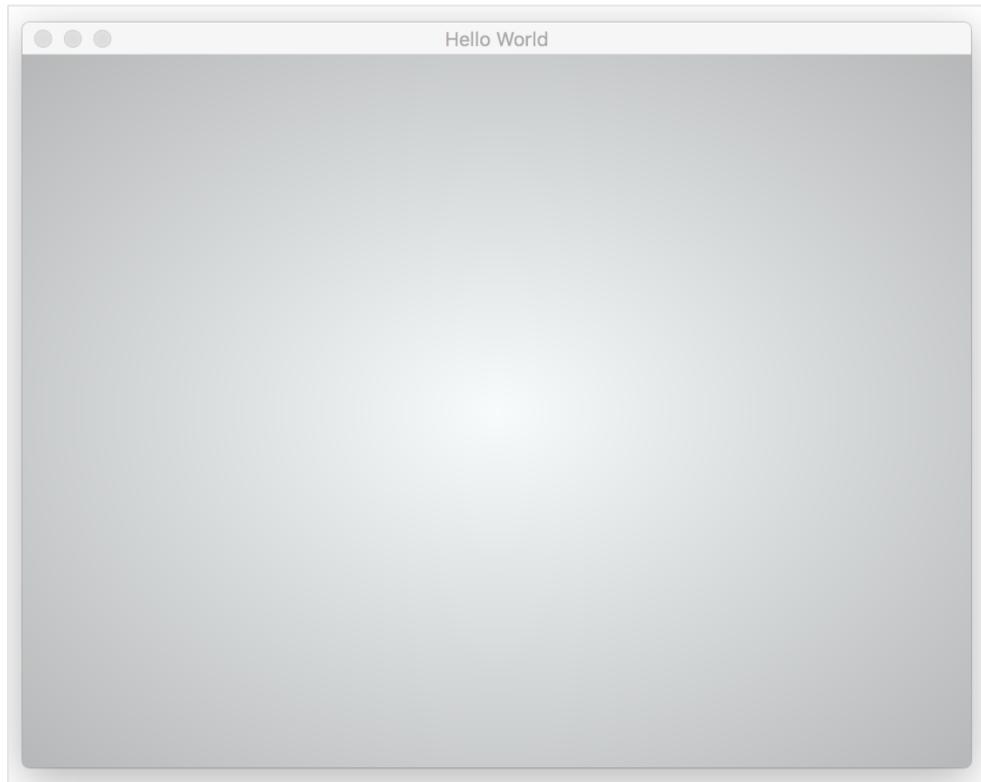
در کد فوق دستور QQuickStyle::setStyle("Material"); تنظیم پوسته را بر روی نوع متریال گوگل اعمال می‌کند. روش‌های مختلفی برای باز طراحی و سفارشی سازی پوسته وجود دارد توسط C++ و QML صورت می‌گیرد. حال با توجه به اینکه قرار است از کنترل‌های موجود در ماژول Qt Quick Control 2.x استفاده شود که تقریباً یک روش کلی برای تمامی آن‌ها وجود دارد. در رابطه با اصلی‌ترین کنترل موجود در کیوت کوئیک کنترل دوم، کنترل ApplicationWindow را توسط مشخصه background بررسی می‌کنیم. مشخصه background جهت اعمال پوسته سفارشی برای پیش‌زمینه کنترل مورد استفاده قرار می‌گیرد. به عنوان مثال کد زیر نشان می‌دهد که چطور می‌توان یک رنگ دلخواه را برای کنترل فراهم ساخت.

```
import QtQuick 2.9
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.3
import QtGraphicalEffects 1.0

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    background: RadialGradient {
        anchors.fill: parent
        gradient: Gradient {
            GradientStop { position: 0.0; color: "#f7fbfc" }
            GradientStop { position: 1.0; color: "#9a9b9c" }
        }
    }
}
```

کد مثال با ترکیب نوع background بر روی مشخصه RadialGradient نتیجه زیر را فراهم خواهد آورد:



ترکیب رنگ بر اساس استاندارد گریدیان که در نرم‌افزارهای فتوشاپ نیز بیشتر مورد استفاده قرار می‌گیرد اینجا تنها با درج کد مربوطه در بازه صورت می‌گیرد که شدت ترکیب رنگ را تحت مقادیر position می‌توان تنظیم کرد.

برای توضیحات بیشتر قرار است پوسته سفارشی برای یک کنترل مانند Button را طراحی و باز نویسی کنیم. در حالت عادی کنترل ما به صورت زیر خواهد بود:

```
Button {
    id: control
    text: qsTr("Button")
}
```



همانطور که مشخص است یک کنترل شامل پس زمینه و رویه (محتو) است. جهت اعمال تغییرات بر روی پس زمینه با مشخصه background و برای تغییرات بر روی رویه کنترل (شیء) از مشخصه contentItem استفاده می‌شود.

```
Button {
    id: control
    text: qsTr("Button")
    anchors.centerIn: parent

    contentItem: Text {
        text: control.text
        font: control.font
        opacity: enabled ? 1.0 : 0.3
        color: control.down ? "#ffffff" : "#f1f1f1"
        horizontalAlignment: Text.AlignHCenter
        verticalAlignment: Text.AlignVCenter
    }
}
```

```

        elide: Text.ElideRight
    }

background: Rectangle {
    implicitWidth: 100
    implicitHeight: 40
    opacity: enabled ? 1 : 0.3
    color: control.down ? "#00aaff" : "#0091ff"
    border.color: control.down ? "#00aaff" : "#0091ff"
    border.width: 1
    radius: 2
}
}

```

با توجه به این که محتوای کنترل از نوع متن است لازم است contentItem را برابر با شیء Text قرار دهیم. در نهایت برای تخصیص ویژگی برای پس زمینه کنترل نیاز است شیء از نوع Rectangle را برابر مشخص نماییم. تمامی ویژگی‌های مربوطه به اشیاء که قبل از مورد آنها اشاره شده است بر روی مشخصه‌ها قابل اعمال است. نتیجه کد فوق به صورت زیر است:



گرینه‌های دیگری در طراحی پوسته وجود دارند مانند CheckBox که در بعضی از آیتم‌ها مانند indicator وجود دارد که کارآیی آن به صورت زیر است:

```

CheckBox {
    id: control
    text: qsTr("CheckBox")
    checked: true

    indicator: Rectangle {
        implicitWidth: 26
        implicitHeight: 26
        x: control.leftPadding
        y: parent.height / 2 - height / 2
        radius: 3
        border.color: control.down ? "#00aaff" : "#0091ff"

        Rectangle {
            width: 14
            height: 14
            x: 6
            y: 6
            radius: 2
            color: control.down ? "#00aaff" : "#0091ff"
            visible: control.checked
        }
    }

    contentItem: Text {
        text: control.text
        font: control.font
        opacity: enabled ? 1.0 : 0.3
        color: control.down ? "#00aaff" : "#0091ff"
        horizontalAlignment: Text.AlignHCenter
    }
}

```

```

        verticalAlignment: Text.AlignVCenter
        leftPadding: control.indicator.width + control.spacing
    }
}

```

همانطور که مشاهده می‌کنید در این نوع جهت اعمال تغییرات از مشخصه indicator استفاده شده است که بیشتر در رابطه با پس زمینه کنترل‌های متغیر مورد استفاده قرار می‌گیرد که کد فوق نتیجه‌ای به صورت زیر را خواهد داشت:



همچنین مواردی در رابطه با delegate و popup وجود دارد که در بعضی از کنترل‌ها از جمله ComboBox مورد استفاده قرار می‌گیرد. کد نمونه آن به صورت زیر آمده است:

```

ComboBox {
    id: control
    anchors.centerIn: parent
    model: ["ONE", "TWO", "THREE"]

    delegate: ItemDelegate {
        width: control.width
        contentItem: Text {
            text: modelData
            color: "#00aaff"
            font: control.font
            elide: Text.ElideRight
            verticalAlignment: Text.AlignVCenter
        }
        highlighted: control.highlightedIndex === index
    }

    indicator: Canvas {
        id: canvas
        x: control.width - width - control.rightPadding
        y: control.topPadding + (control.availableHeight - height) / 2
        width: 12
        height: 8
        contextType: "2d"

        Connections {
            target: control
            onPressedChanged: canvas.requestPaint()
        }

        onPaint: {
            context.reset();
            context.moveTo(0, 0);
            context.lineTo(width, 0);
            context.lineTo(width / 2, height);
            context.closePath();
            context.fillStyle = control.pressed ? "#17a81a" : "#00aaff";
            context.fill();
        }
    }

    contentItem: Text {
        leftPadding: 10
        rightPadding: control.indicator.width + control.spacing
    }
}

```

```

text: control.displayText
font: control.font
color: control.pressed ? "#17a81a" : "#00aaff"
horizontalAlignment: Text.AlignLeft
verticalAlignment: Text.AlignVCenter
elide: Text.ElideRight
}

background: Rectangle {
    implicitWidth: 120
    implicitHeight: 40
    border.color: control.pressed ? "#17a81a" : "#00aaff"
    border.width: control.visualFocus ? 2 : 1
    radius: 2
}

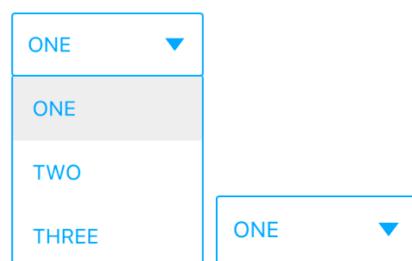
popup: Popup {
    y: control.height - 1
    width: control.width
    implicitHeight: contentItem.implicitHeight
    padding: 1

    contentItem: ListView {
        clip: true
        implicitHeight: contentHeight
        model: control.popup.visible ? control.delegateModel : null
        currentIndex: control.highlightedIndex

        ScrollIndicator.vertical: ScrollIndicator { }
    }

    background: Rectangle {
        border.color: "#00aaff"
        radius: 2
    }
}
}

```



مثالی دیگر از نوع کنترل DelayButton به صورت زیر خواهد بود:

```

DelayButton {
    id: control
    checked: true
    text: qsTr("Delay\nButton")

    contentItem: Text {
        text: control.text
        font: control.font
        opacity: enabled ? 1.0 : 0.3
    }
}

```

```
        color: "white"
        horizontalAlignment: Text.AlignHCenter
        verticalAlignment: Text.AlignVCenter
        elide: Text.ElideRight
    }

background: Rectangle {
    implicitWidth: 100
    implicitHeight: 100
    opacity: enabled ? 1 : 0.3
    color: control.down ? "#00bbff" : "#00aaff"
    radius: size / 2

    readonly property real size: Math.min(control.width, control.height)
    width: size
    height: size
    anchors.centerIn: parent

    Canvas {
        id: canvas
        anchors.fill: parent

        Connections {
            target: control
            onProgressChanged: canvas.requestPaint()
        }

        onPaint: {
            var ctx = getContext("2d")
            ctx.clearRect(0, 0, width, height)
            ctx.strokeStyle = "white"
            ctx.lineWidth = parent.size / 20
            ctx.beginPath()
            var startAngle = Math.PI / 5 * 3
            var endAngle = startAngle + control.progress * Math.PI / 5 * 9
            ctx.arc(width / 2, height / 2, width / 2 - ctx.lineWidth / 2 - 2, startAngle, endAngle)
            ctx.stroke()
        }
    }
}
```

با توجه به اعمال ویژگی بر روی انواع کنترل‌ها توسط مشخصه‌های `background`, `contentItem` و ... نتیجه نهایی آن‌ها بر اساس مقادیری که برای آن‌ها تعریف می‌شوند اعمال و نمایان می‌گردد.



مثالی دیگر در رابطه با کنترل پرکاربرد در صنایع مختلف Dial است که برای سفارشی سازی آن نمونه‌ای به صورت زیر خواهیم داشت:

Dial {

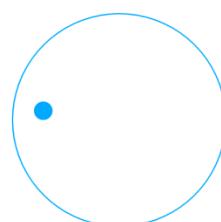
```

id: control
background: Rectangle {
    x: control.width / 2 - width / 2
    y: control.height / 2 - height / 2
    width: Math.max(64, Math.min(control.width, control.height))
    height: width
    color: "transparent"
    radius: width / 2
    border.color: control.pressed ? "#00bbff" : "#00aaff"
    opacity: control.enabled ? 1 : 0.3
}

handle: Rectangle {
    id: handleItem
    x: control.background.x + control.background.width / 2 - width / 2
    y: control.background.y + control.background.height / 2 - height / 2
    width: 16
    height: 16
    color: control.pressed ? "#00bbff" : "#00aaff"
    radius: 8
    antialiasing: true
    opacity: control.enabled ? 1 : 0.3
    transform: [
        Translate {
            y: -Math.min(control.background.width,
control.background.height) * 0.4 + handleItem.height / 2
        },
        Rotation {
            angle: control.angle
            origin.x: handleItem.width / 2
            origin.y: handleItem.height / 2
        }
    ]
}
}

```

مشخصه handle یک خصیصه ویژه در کنترل‌های خاص محسوب می‌شود که عمل نگه دارنده و برجستگی بر روی کنترل را جهت تنظیم مقدار و ارزش آن مشخص می‌کند که به راحتی با نوع شیء Rectangle به صورت سفارشی باز طراحی شده است.



طراحی بر روی کنترل‌ها تقریباً طبق مثال‌های ارائه شده به طور یکسان است، این بدین معنی که از مشخصه‌های background.indicator و غیره بر اساس صفت‌هایی که در صفحات قبلی راجع به انواع کنترل توضیح داده شده است تبعیت می‌کند.

## واکنش گرایی - مقیاس پذیر بودن (Scalability)

با توجه به یکی از نکات مهم تجربه کاربری (UX) واکنش گرا بودن یک محصول به اندازه خود محصول مهم است، بنابراین اگر قرار باشد محصولی را در دستگاه‌های مختلف ارائه دهیم مسلماً باید نسبت به نوع دستگاه و صفحه نمایش آن تنظیم و پیکربندی شود که این کار باید به صورت خودکار بر روی نرم‌افزار تعییه شده باشد. به عنوان مثال صفحه نمایش گوشی‌های هوشمند با پلتفرم دسکتاپ متفاوت است برای اینکه محیط نرم‌افزار بر روی هر یک از این تجهیزات بهینه و واکنشگرا شود لازم است این مبحث را رعایت نماییم.

به طور پیشفرض کنترل Quick layout این امکان را فراهم می‌سازد که آیتم‌های شما بر اساس ویژگی‌هایی که دارند بر روی فرم ظاهر و واکنش پذیر باشند. در ادامه کد نمونه‌ای از نحوه عملکرد این امکان آورده شده است.

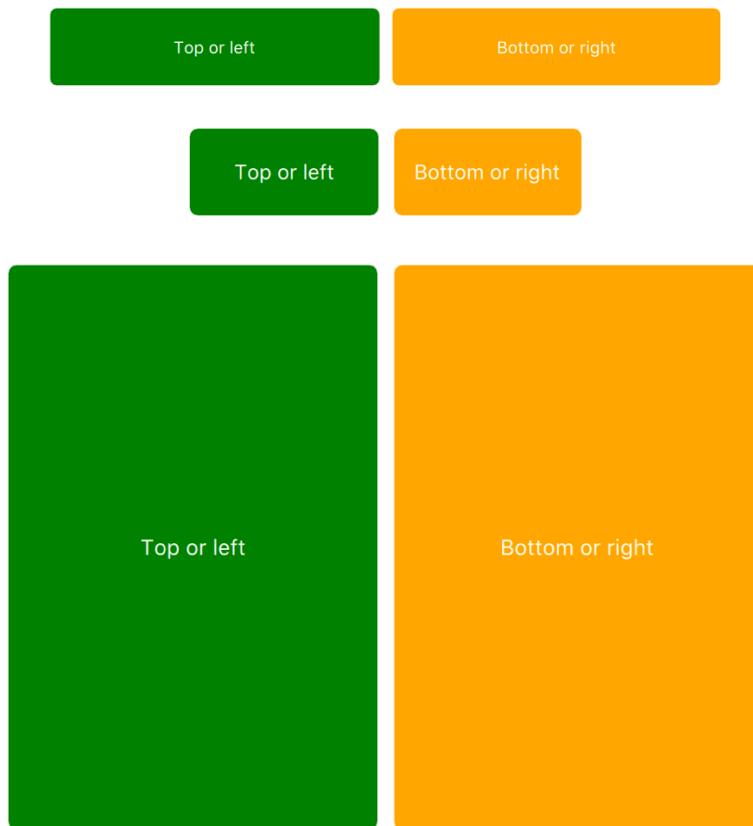
```
GridLayout {
    anchors.fill: parent
    anchors.margins: 10
    rowSpacing: 10
    columnSpacing: 10
    flow: width > height ? GridLayout.LeftToRight : GridLayout.TopToBottom
    Rectangle {
```

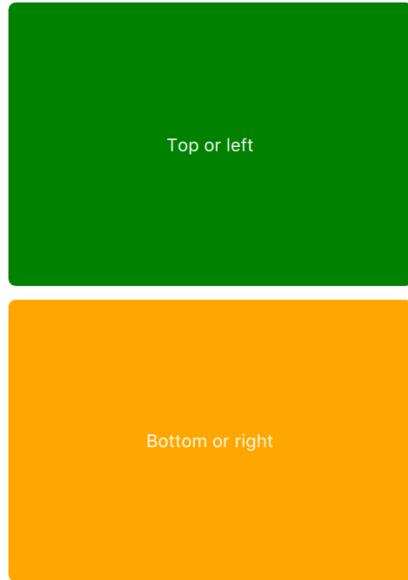
```

Layout.fillWidth: true
Layout.fillHeight: true
radius: 5
color: "green"
Label {
    anchors.centerIn: parent
    text: "Top or left"
    color: "white"
}
}
Rectangle {
    Layout.fillWidth: true
    Layout.fillHeight: true
    radius: 5
    color: "orange"
    Label {
        anchors.centerIn: parent
        text: "Bottom or right"
        color: "white"
    }
}
}

```

با توجه به کد فوق در زمان تغییر ابعاد شیء Rectangle بر روی فرم به شیوه‌های زیر تغییر و اکنش نشان خواهد داد.





موارد فوق با تغییر اندازه صفحه نمایش و یا تغییر ابعاد بر روی صفحه بر اساس نوع دستگاه تغییر می‌یابند. که دستور موثر در کد مربوطه به صورت زیر است:

```
flow: width > height ? GridLayout.LeftToRight : GridLayout.TopToBottom
```

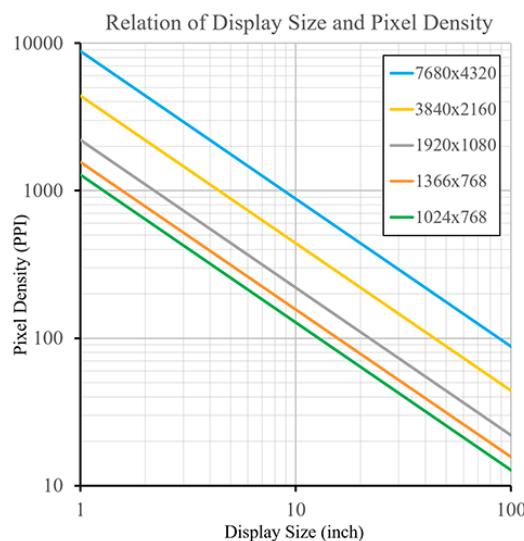
دستور فوق شرط بر اندازه‌های ارتفاع و طول فرم فراهم می‌کند که بر اساس آن نوع چیدمان بر روی GridLayout را تغییر می‌دهد از بالا به پایین و از چپ به راست نحوه چیدمان آیتم‌های موجود بر روی GridLayout تنظیم می‌شوند.

نکته قابل توجهی که در این مبحث وجود دارد این است که در صورت عدم تغییر ابعاد و ایجاد خاصیت واکنش‌گرایی در فرم و آیتم‌ها امکان کاهش کارآیی و حتی سرعت (فریم ریت - FPS) در پردازش ممکن است ایجاد شود. چرا که در هر بار تغییر سیستم به طور خودکار مقیاس را بر اساس ابعاد حفظ می‌کند و این در حفظ کیفیت نیز موثر است و ممکن است در کاهش انرژی باطری بر روی گوشی و یا سیستم‌های فاقد برق مستقیم تاثیر بسیاری داشه باشد. برای اینکه ما کاری انجام دهیم تا سیستم همه بار پردازشی را بر عهده پردازنده قرار ندهد بهتر است از قبل از آن به مواردی باید توجه داشته باشیم که در موقع استفاده از Layout‌ها در نظر داشته باشیم:

- خاصیت‌های محورهای x، y و حتی width و height را در موقع استفاده به نوع Layout متصل نکنید چرا که این امر باعث ایجاد تضاد در مقاصد لایه Layout باشد و ممکن است یک حلقه اتصال ایجاد کند که باعث تخریب و عدم ناهمانگی ایجاد کند.
- دقت داشته باشید که از توابع پیچیده جاوا اسکریپتی (JavaScript) که مدام مورد مقایسه و بررسی قرار می‌گیرند استفاده نکنید. این امر عملکرد باعث کاهش کارآیی شده و مخصوصاً در مباحث انیمیشن و جلوه‌های خاص بسیار تاثیر گذار است.
- به طور فرضی مقایسه در رابطه با اندازه نگه دارنده‌ها یا همان Layout‌ها انجام ندهید حتی در مورد اندازه‌ایتم‌های فرزند آنها هیچ نوع مقایسه فرضی نباید صورت بگیرد. به طور کلی سعی بر این داشته باشید که یک تعریف انعطاف پذیر را انجام دهید که بتواند در یک فضای موجودی که در اختیار دارد واکنش نشان دهد.
- اگر طرح شما قرار است یک طرح کامل پیکسلی باشد و کیفیت خودش را تضمین کند به هیچ عنوان از لایه‌ها استفاده نکنید. محتوای انواع اشیاء می‌توانند به طور خودکار بر اساس فضای موجودی که در اطرافشان است تغییر اندازه دهند.

در رابطه با این موارد امکان اينکه شما بتوانيد برنامه خود را طوري طراحي کنيد که در پلتفرمهاي مانند (iOS, macOS) که امكان تغيير چگالي در پикسل کمتر و بيشتر را به طور خودكار ندارند را فراهم کنيد. با استفاده از **Screen.PixelDensity** می‌توان شيء را برای نمایش در صفحه‌های نمایش با چگالی کم یا بيشتر را مشخص کرد.

قبل از هر چيز نياز است در رابطه با **PixelDensity** اطلاعاتي کسب کنيم. به طور کلي پيكسل در هر اينچ با مخفف (**PPI**) از کلمه **inch** و يا پيكسل بر هر سانتي متر (**PPCM**) مشخص می‌شود که تراکم پيكس يا همان (ريزولوشن - Resolution) را ببررسی می‌کند.



نحوه محاسبه PPI در مانیتورهای مختلف بر اساس قضیه فیثاغورس توسط مقادیر `width` و `height` ممکن است.

$$PPI = \frac{d_p}{d_i}$$

عنوانين زير معرف انواع مقادير هستند:

$d_i$  تفکیک قطری در پيكسل  $h_p$  وضوح عرض در پيكسل  $w_p$  وضوح ارتفاع در پيكسل  $d_p$  تفکیک قطری در اينچ می‌باشند. به عنوان مثال برای محاسبه  $21.5$  اينچ ( $54.61$  سانتي متر) در يك صفحه  $1920 \times 1080$  در  $1080$  پيكسلی فرمولی به صورت زير خواهيم داشت:

$$w_p = 1920, d_i = 21.5 \quad h_p = 1080 \text{ and}$$

مقدار بازگشتی از اين محاسبه  $0.46$  **PPI** خواهد بود.

با توجه به دستگاههای مجهر به **iOS** و **macOS** که خود قابلیت مقیاس‌پذیری را بر روی تصویر ارائه نمی‌دهند مثالی در زیر آورده شده است که برای حل اين مسئله از روش **PixelDensity** استفاده شده است، البته بر روی دستگاههای **Windows** و **Android** چنین مشکلی وجود ندارد.

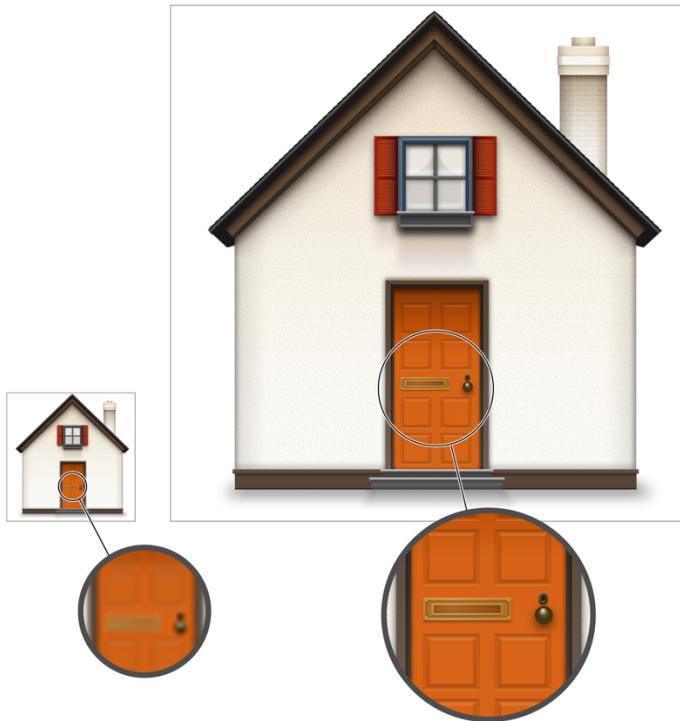
```
Image {
  source: {
    if (Screen.PixelDensity < 40)
      "qrc:/image_low_dpi.png"
    else if (Screen.PixelDensity > 300)
      "qrc:/image_hight_dpi.png"
```

```

    else
        "qrc:/image.png"
}
}

```

بنابراین در صورتی که مقدار ppi صفحه پایین باشن بهتر است از نوع تصویر با منبع عادی استفاده شود در غیر اینصورت دو برابر اندازه موجود در پیکسل مناسب برای تراکم پیکسل‌های بالا است.



مقدار از نوع ثابت `devicePixelRatio` نسبت بین پیکسل‌های فیزیکی و مستقل از دستگاه برای صفحه نمایش را بر می‌گرداند. ارزش‌های مشترک ۱.۰ بر روی صفحه‌های نمایش معمولی و ۲.۰ بر روی صفحه‌های نمایش رتینا (Retina) و مقادیر بالاتر امکان‌پذیر است. که دسترسی به آن توسط کلاس `QScreen` امکان‌پذیر می‌شود که در زیر مثالی از این روش آورده شده است که با فراخوانی سرآیند `QScreen` است:

```
#include <QScreen>
```

کد زیر نمونه‌ای از روش C++ جهت دریافت مقدار DPR است که با مقایسه آن تحت یک دستور شرطی سیستم خود تصویر مورد نظر با کیفیت مناسب برای صفحه نمایش را فراهم می‌کند.

```

if ( QGuiApplication::primaryScreen()->devicePixelRatio() >= 2 ) {
    QVariant imageVariant = "@2x";
    qDebug() << imageVariant;
} else {
    QVariant imageVariant = "";
    qDebug() << imageVariant;
}

```

در ادامه مثالی در رابطه با دریافت مقادیر صفحه و اطلاعات آن پرداخته شده است که کد QML آن به صورت زیر خواهد بود:

```

import QtQuick 2.9
import QtQuick.Window 2.2

Window {
    visible: true
    title: qsTr("Hello World")

    property int margin: 11
    width: 600
    height: 500

    Item {
        id: root
        width: 400
        height: propertyGrid.implicitHeight + 16

        function orientationToString(o) {
            switch (o) {
                case Qt.PrimaryOrientation:
                    return "primary";
                case Qt.PortraitOrientation:
                    return "portrait";
                case Qt.LandscapeOrientation:
                    return "landscape";
                case Qt.InvertedPortraitOrientation:
                    return "inverted portrait";
                case Qt.InvertedLandscapeOrientation:
                    return "inverted landscape";
            }
            return "unknown";
        }

        Grid {
            id: propertyGrid
            columns: 2
            spacing: 8

            x: spacing
            y: spacing

            Text {
                text: "Screen \"\" + Screen.name + ":" +
                font.bold: true
            }

            Item { width: 2; height: 2 }

            Text { text: "Dimensions"; font.bold: true }
            Text { text: Screen.width + "x" + Screen.height }

            Text { text: "Pixel density"; font.bold: true }
            Text { text: Screen.pixelDensity.toFixed(2) + " dots/mm (" +
(Screen.pixelDensity * 25.4).toFixed(2) + " dots/inch)" }

            Text { text: "Logical pixel density"; font.bold: true }
            Text { text: Screen.logicalPixelDensity.toFixed(2) + " dots/mm (" +
(Screen.logicalPixelDensity * 25.4).toFixed(2) + " dots/inch)" }

            Text { text: "Device pixel ratio"; font.bold: true }
            Text { text: Screen.devicePixelRatio.toFixed(2) }
        }
    }
}

```

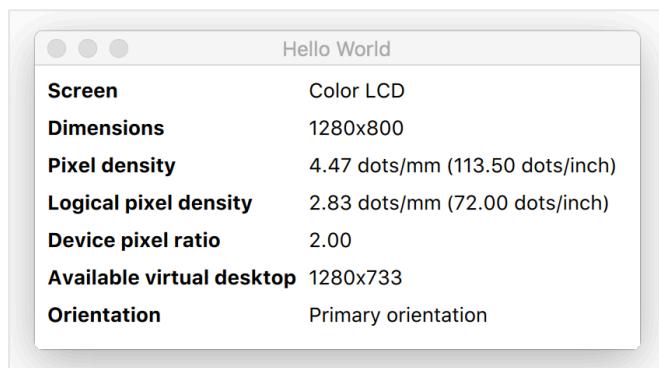
```

Text { text: "Available virtual desktop"; font.bold: true }
Text { text: Screen.desktopAvailableWidth + "x" +
Screen.desktopAvailableHeight }

Text { text: "Orientation"; font.bold: true }
Text { text: orientationToString(Screen.orientation) + " (" +
Screen.orientation + ")" }

Text { text: "Primary orientation"; }
Text { text: orientationToString(Screen.primaryOrientation) + " (" +
Screen.primaryOrientation + ")" }
}
}
}

```



با توجه به نوع کد در QML برای دسترسی به کلاس Screen تحت QScreen نیاز است ماتریس Window را فراخوانی نماییم که به صورت زیر است.

```
import QtQuick.Window 2.2
```

مقادیر name جهت دریافت نام صفحه نمایش و مقادیری از جمله width و height جهت دریافت مقادیر طول و عرض صفحه است و همچنین طبق توضیحی که داده شد pixelDeviceRatio مقدار چگالی و مشخصه pixelDensity مقدار مربوط به نسبت پیکسل را برمی‌گرداند. همچنین مقدار طول مجازی بر روی صفحه توسط مشخصه orientationToString و desktopAvailableWidth مقادیر شمارشی وضعیت چرخشی صفحه را ارائه می‌کند.

ثبتات	ارزش	توضیحات
Qt::PrimaryOrientation	0x00000000	گرایش یا جهت صفحه نمایش اصلی را برمی‌گرداند.
Qt::LandscapeOrientation	0x00000002	نوع چشم انداز و جهت صفحه نمایش را برای طول بیشتر از عرض آن تعیین می‌کند.
Qt::PortraitOrientation	0x00000001	نوع چشم انداز و جهت صفحه نمایش را به صورت ۹۰ درجه در جهت عقربه‌های ساعت تغییر می‌دهد.
Qt::InvertedLandscapeOrientation	0x00000008	معکوس جهت افقی را در بازه ۱۸۰ درجه نمای می‌دهد.
Qt::InvertedPortraitOrientation	0x00000004	معکوس جهت عمودی را در بازه ۱۸۰ درجه نمای می‌دهد.

```
#include <QGuiApplication>
#include <QScreen>
#include <QDebug>

/*
 Example of using Qt 5 QScreen class.
 */

// Helper function to return display orientation as a string.
QString Orientation(Qt::ScreenOrientation orientation)
{
    switch (orientation) {
        case Qt::PrimaryOrientation : return "Primary";
        case Qt::LandscapeOrientation : return "Landscape";
        case Qt::PortraitOrientation : return "Portrait";
        case Qt::InvertedLandscapeOrientation : return "Inverted landscape";
        case Qt::InvertedPortraitOrientation : return "Inverted portrait";
        default : return "Unknown";
    }
}

int main(int argc, char *argv[])
{
    QGuiApplication a(argc, argv);

    qDebug() << "Number of screens:" << QGuiApplication::screens().size();

    qDebug() << "Primary screen:" << QGuiApplication::primaryScreen()->name();

    foreach (QScreen *screen, QGuiApplication::screens()) {
        qDebug() << "Information for screen:" << screen->name();
        qDebug() << " Available geometry:" << screen->availableGeometry().x() <<
screen->availableGeometry().y() << screen->availableGeometry().width() << "x" <<
screen->availableGeometry().height();
        qDebug() << " Available size:" << screen->availableSize().width() << "x"
<< screen->availableSize().height();
        qDebug() << " Available virtual geometry:" << screen-
>availableVirtualGeometry().x() << screen->availableVirtualGeometry().y() <<
screen->availableVirtualGeometry().width() << "x" << screen-
>availableVirtualGeometry().height();
        qDebug() << " Available virtual size:" << screen-
>availableVirtualSize().width() << "x" << screen->availableVirtualSize().height();
        qDebug() << " Depth:" << screen->depth() << "bits";
        qDebug() << " Geometry:" << screen->geometry().x() << screen-
>geometry().y() << screen->geometry().width() << "x" << screen-
>geometry().height();
        qDebug() << " Logical DPI:" << screen->logicalDotsPerInch();
        qDebug() << " Logical DPI X:" << screen->logicalDotsPerInchX();
        qDebug() << " Logical DPI Y:" << screen->logicalDotsPerInchY();
        qDebug() << " Orientation:" << Orientation(screen->orientation());
        qDebug() << " Physical DPI:" << screen->physicalDotsPerInch();
        qDebug() << " Physical DPI X:" << screen->physicalDotsPerInchX();
        qDebug() << " Physical DPI Y:" << screen->physicalDotsPerInchY();
        qDebug() << " Physical size:" << screen->physicalSize().width() << "x" <<
screen->physicalSize().height() << "mm";
        qDebug() << " Primary orientation:" << Orientation(screen-
>primaryOrientation());
        qDebug() << " Refresh rate:" << screen->refreshRate() << "Hz";
    }
}
```

```

    qDebug() << "  Size:" << screen->size().width() << "x" << screen-
>size().height();
    qDebug() << "  Virtual geometry:" << screen->virtualGeometry().x() <<
screen->virtualGeometry().y() << screen->virtualGeometry().width() << "x" <<
screen->virtualGeometry().height();
    qDebug() << "  Virtual size:" << screen->virtualSize().width() << "x" <<
screen->virtualSize().height();
}
}

```

خروجی کد فوق به صورت زیر خواهد بود:

```

Number of screens: 1
Primary screen: "Color LCD"
Information for screen: "Color LCD"
Available geometry: 0 23 1280 x 732
Available size: 1280 x 732
Available virtual geometry: 0 23 1280 x 732
Available virtual size: 1280 x 732
Depth: 24 bits
Geometry: 0 0 1280 x 800
Logical DPI: 72
Logical DPI X: 72
Logical DPI Y: 72
Orientation: "Landscape"
Physical DPI: 113.5
Physical DPI X: 113.5
Physical DPI Y: 113.5
Physical size: 286.449 x 179.031 mm
Primary orientation: "Landscape"
Refresh rate: 60 Hz
Size: 1280 x 800
Virtual geometry: 0 0 1280 x 800
Virtual size: 1280 x 800

```

مقادیر فوق در زمان دریافت اطلاعات صفحه جهت تعریف مقادیر سفارشی بسیار کار آمد خواهند بود. برای مثال با دریافت Dpi به راحتی می‌توان کد دستورات شرطی را مقدار دهی و بررسی کرد. هرچند استاندارد هر یک مشخص است اما در این مثال به مقادیر بازگشتی کلاس QScreen اشاره شده است که برای طراحان UX/UI مورد نیاز است.

## محتوای وب در برنامه توسط Qt WebEngine

برای طراحی برنامه بر پایه وب کیوت فناوری‌های مرتبط با وب مانند HTML, CSS و JavaScript را توسط ماژول WebEngine فراهم می‌کند. برنامه‌های شما تحت فناوری وب طراحی خواهند شد و به طور کلی ترکیب کدهای QML و C++ با کدهای HTML و JavaScript برای طراحی برنامه‌های وب ممکن است.

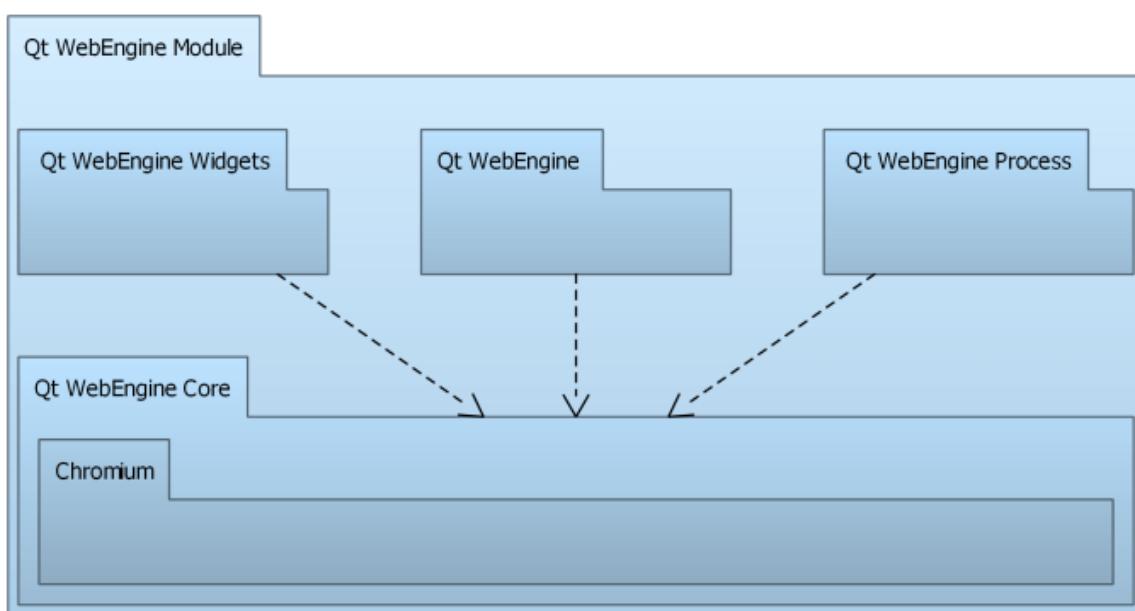
تعریف کلی ماژول Qt Web Engine این است که این ماژول بر پایه موتور کرومیوم (Chromium) گوگل و هسته V8 انجین توسعه یافته که برای اهداف طراحی اپلیکیشن در پلتفرم‌های دسکتاپ و ایمبدہا مورد استفاده قرار می‌گیرد. برای مثال توسط این ماژول می‌توان هر آنچه را که برای ساخت یک مرورگر برای پلتفرمی مانند لینوکس نیاز است را در اختیار داشت. البته کیوت ماژولی را با نام Qt WebView مخصوص پلتفرم‌های موبایل ارائه داده است که بیشتر برای بخش QML تحت Qt Quick است.

به طور کلی Qt WebEngine کلاس‌هایی را در C++ و انواعی را در QML ارائه داده است که امکان ساخت استانداردهای HTML، XHTML، JavaScript و SVG را فراهم می‌کند که طراحی پوسته آن نیز توسط CSS و اسکریپت نویسی آن توسط HTML ممکن است. البته استناد HTML به طور کامل قابل ویرایش از سمت کاربر می‌باشد چرا که محتوا ویژگی قابل ویرایش عناصر را در خود دارد.

## معماری ماژول

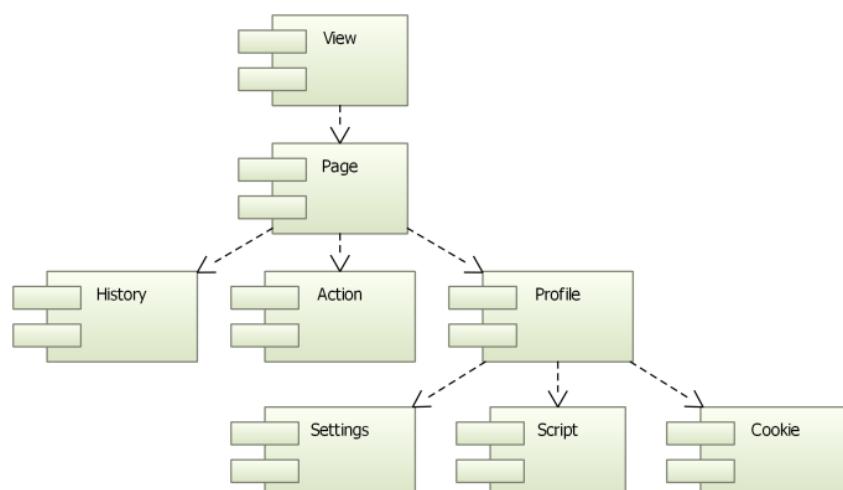
البته باید توجه داشته باشیم که قابلیتها در این ماژول به سه ماژول زیر تقسیم بندی شده است:

۱. ماژول (Qt WebEngine Widgets Module) جهت ساخت اپلیکیشن بر پایه وب و ویجت‌ها (QWidget) سبک سنتی
۲. ماژول (Qt WebEngine Module) جهت ساخت اپلیکیشن بر پایه فناوری کیوت کوئیک (Qt Quick) وب
۳. ماژول (Qt WebEngine Core Module) جهت هماهنگی و یکپارچه سازی با هسته کرومیوم



نکته: تولید صفحات و اجرای کدهای جاوا اسکریپتی در بخش گرافیکی (GUI) از هسته پردازشی آن Qt Web Engine Process جدا شده است.

## معرفی ماژول (Qt WebEngine Widgets Module)

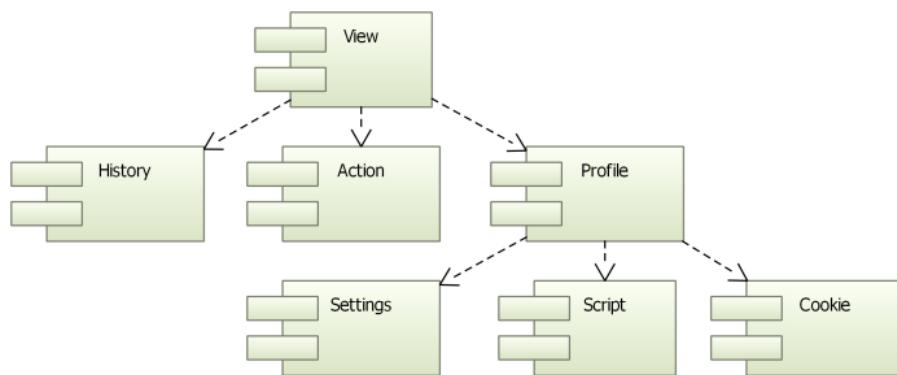


طبق توضیحات مربوط به معماری وب انجین یک نما (view) در انجین وب کیوت جزء اصلی از یک ماژول Qt Web Engine بشمار می‌رود که می‌توان از آن در برنامه‌های مختلفی جهت بارگذاری محتوای وب استفاده کرد. در یک نگاه یک صفحه (Page) در وب انجین قاب اصلی نگه دارنده محتوای وب مانند تاریخچه (History)، لینک‌ها، تنظیمات، کوکی‌ها و عملیات متفاوتی دیگری است.

تمامی این صفحات متعلق به یک پروفایل از موارد به اشتراک گذاری شده مانند، تنظیمات، اسکریپتها و کوکی‌ها است. پروفایل را می‌توان برای جدا سازی صفحات از یکدیگر مورد استفاده قرار داد. حالت معمولی در استفاده از آن یک پروفایل اختصاصی جهت مرورگری در حالت خصوصی است، یعنی هیچ اطلاعاتی از آن برای مدت دائمی ذخیره نمی‌شود.

توجه داشته باشید که ماژول Qt Web Engine Widgets از نمودار صحنه‌ای فناوری Qt Quick برای تشکیل عناصر از یک صفحه وب را در مدل نما (view) استفاده می‌کند. این بدین معنی است که پردازش نیازمند OpenGL ES 2.0 و یا OpenGL 2.0 برای تولید است.

### معرفی ماژول (Qt WebEngine Module)



ماژول Qt Web Engine شامل همان عناصری است که در ماژول Qt Web Engine Widgets موجود هستند، به جز اینکه دسترسی به صفحات وب انجین جدا از هم است. پشتیبانی از یکپارچه بودن نمایه (view) و عملیات به صورت یکپارچه فراهم شده است.

### معرفی ماژول (Qt WebEngine Core)

هسته Qt WebEngine بر پایه پروژه کرومیوم است. پروژه کرومیوم همانطور که مشخص شده است موتور اختصاصی شبکه و رسم صحنه خود را دارد که با ماژول‌های وابسته به آن توسعه یافته‌اند. به این نکته نیز توجه داشته باشید که هرچند این ماژول بر پایه کروم است اما این بدین معنی نیست که برخی از سرویس‌های مرورگر کروم را پشتیبانی می‌کند که گوگل برای آن فراهم کرده است. بنابراین باید توجه داشته باشیم که کرومیوم نام اختصاص داده شده به یک پروژه منبع باز است که کد منبع آن توسط پروژه کرومیوم منتشر و نگهداری می‌شود. نصب آخرین نسخه‌های طراحی شده توسط رونوشت‌های از پیش کامپایل شده برای ویندوز، لینوکس و مک او اس امکان‌پذیر است، گوگل کد منبع کرومیوم را با اضافه کردن یک فلش پلیر داخلی، نشان گوگل، یک بروزرسان خودکار به نام بروزرسان گوگل (GoogleUpdate)، یک آمارگیر (که به سلیقه شخص، اطلاعات مربوط به شکست‌ها و آمار نحوه عملکرد مرورگر را برای گوگل ارسال می‌کند) تحت عنوان کروم منتشر می‌کند.

نسخه کنونی وب انجین کیوت در نسخه ۵.۱۵ کتابخانه، با برخی از بهروزرسانی‌های امنیتی همراه است.

اما در مورد Qt Web Engine Process باید اینگونه توضیح دهیم که یک فایل اجرا کننده مجزایی از پروژه است که برای تولید صفحات وب و اجرای کدهای جاوااسکریپتی مورد استفاده قرار می‌گیرد. این امر باعث می‌شود تا مسائل امنیتی و ناشی از محتوای خاص کاهش یابند.

جهت دسترسی به این ماژول طبق روال نام ماژول مربوطه را در فایل pro اضافه خواهیم کرد بدین صورت:

```
QT += webengine
```

جهت نمایش محتوای وب در حالت عادی از کلاس QWebView استفاده خواهیم کرد این ساده‌ترین راه ممکن است. زیرا آن خود نیز یک ویجت بشمار می‌رود. بنابراین مثال فوق را خواهیم داشت.

```
#include <QApplication>
#include <QWebEngineView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QWebEngineView *view = new QWebEngineView(nullptr);
    view->load(QUrl("http://www.genyleap.com/"));
    view->show();

    return app.exec();
}
```

در کد فوق قبل از استفاده از کلاس QWebView فایل سرآیند آن را فراخوانی و سپس از کلاس مربوط به آن یک شیء جدید خواهیم ساخت. در نهایت با فراخوانی تابع load() که ساختار زیر را دارد آدرس مورد نظر را برای فراخوانی ارسال می‌کنیم. تنها باید توجه داشته باشید که مقدار ورودی از نوع رشته و یک آدرس ثابت خواهد بود.

```
void QWebEngineView::load(const QUrl &url)
```

سپس تابع show جهت نمایش در آوردن مدل اصلی خواهد بود که همان تابع show برای تمامی ویجت‌ها است. در نهایت نتیجه آن به صورت یک صفحه وب که دارای محتوای موجود در آدرس است نمایان خواهد شد.

نمونه‌ای از یک QWebEngineView یک QWebEnginePage را دارا است که خود شامل یک QWebEngineHistory است که امکان دسترسی به سوابق پیمایش شده و چند نوع از QAction را در صفحه فراهم می‌کند. علاوه بر این، یک QWebEnginePage قادر به اجرای کد ای جاوااسکریپتی است که برای رویدادهایی مانند نمایش دیالوگ (پنجره)‌های پرسش به عنوان مثال نام کاربری و رمزعبور است.

برای برنامه‌هایی که بر پایه ویجت هستند، در این انواع وب انجین کیوت به صورت خودکار مقدار دهی اولیه را برای ماژول انجام می‌دهد. مگر اینکه آن را به صورت یک پلاگین قرار داده باشید که در این صورت، باید در سورس اصلی با استفاده از متدهای QtWebEngine::initialize مقدار دهی اولیه شود مانند کد زیر:

```
#include <QApplication>
#include <QtWidgets>
#include <QtWebEngine>

int main(int argc, char **argv)
{
    QApplication app(argc, argv);
```

```

QtWebEngine::initialize();

QMainWindow window;
window.show();

return app.exec();
}

```

در رابطه با انواع پروژه‌ها بر پایه فناوري کیوت کوئیک این امکان وجود دارد که به طور پویا محتواي وب را در انواع قسمت‌ها نمایش دهد بنابراین برای مقدار دهی اولیه در نوع اپلیکیشن Quick Qt به صورت زیر عمل خواهیم کرد:

```

#include <QApplication>
#include <QQmlApplicationEngine>
#include <QtWebEngine>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);

    QtWebEngine::initialize();

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}

```

حال با فرض اینکه از فناوري Quick جهت نمایش محتوا استفاده خواهیم کرد کد QML برای نمونه به صورت زیر خواهد بود که توسط WebViewEngine فراهم می‌شود.

```

import QtQuick 2.9
import QtQuick.Window 2.2
import QtWebEngine 1.3

Window {
    width: 1024
    height: 750
    visible: true

    WebEngineView {
        anchors.fill: parent
        url: "http://www.genyleap.com"
    }
}

```

تعاریف فوق مرتبط با معرفی ماژول QtWebEngine بود که نشان می‌دهد چگونه توسط این ماژول می‌توان به پیش نیازات محتواي تحت وب دسترسی داشت. نیاز است در رابطه با ترکیب محتواي وب با اپلیکیشن تحت C++ و QML توضیح و مثالی بزنیم.

فرض کنید نیاز است برنامه‌ای طراحی شود که در آن با استفاده از سرویس گوگل مپ به امکان نقشه گوگل دسترسی داشته باشیم که بخشی از آن نیاز است محتواي وب باشد. بنابراین برای این کار کدهای C++ و QML را همراه با JavaScript و HTML ترکیب خواهیم کرد.

ابتدا پروژه‌ای را از نوع Qt Quick ایجاد کرده و سپس فایل‌های map.h و map.cpp را با محتوای سفارشی خواهیم ساخت.  
فایل مربوط به map.h دارای کدهای زیر خواهد بود:

```
#ifndef MAP_H
#define MAP_H

#include <QObject>
#include <QString>

class Map : public QObject
{
    Q_OBJECT

public:
    Map();

    Q_INVOKABLE QString key( const QString &k );
    Q_INVOKABLE float lat( const float &x );
    Q_INVOKABLE float lng( const float &y );
    Q_INVOKABLE float zoom( const float &val );

};

#endif // MAP_H
```

کلاسی بر پایه QObject ساخته و سپس توابع مورد نیاز را تعریف می‌کنیم. در این مثال ما نیاز به دریافت مقادیر مرتبط با محور x، y و متغیر zoom هستیم. البته بر اساس ساختار API در گوگل تابعی برای نگه‌داری متغیر key یا همان شناسه توسعه دهنده ایجاد می‌کنیم، دقت کنید که برای اینکه نیاز است مقادیر هریک از توابع در QML مورد استفاده بگیرد کلمه کلیدی Q\_INVOKABLE اجبار خواهد بود.

محتوای مربوط به فایل map.cpp به صورت زیر خواهد بود:

```
#include "map.h"

Map::Map() { }

QString Map::key( const QString &k ) {
    return k;
}

float Map::lat( const float &x ) {
    return x;
}

float Map::lng( const float &y ) {
    return y;
}

float Map::zoom( const float &val ) {
    return val;
}
```

بدنه هر یک از توابع را با مقدار بازگشتی پارامترهای هریک ایجاد می‌کنیم. سپس در تابع main کد زیر را خواهیم داشت که عمل ثبت کلاس سی‌پلاس‌پلاس را برای QML انجام می‌دهد:

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QtQml>
#include <QtWebEngine>

//Include My map class
#include "map.h"

int main(int argc, char *argv[])
{
    QCOREAPPLICATION::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);

    QtWebEngine::initialize(); //Initialize WebEngine!

    qmlRegisterType<Map>("Interface.Genyleap", 1, 0, "Map");

    QQmlApplicationEngine engine;
    engine.load(QUrl(QLatin1String("qrc:/main.qml")));

    return app.exec();
}
```

کلاس Map را توسط دستور اینکلود وارد کرده و آن را توسط qmlRegisterType که قبلا در مورد آن توضیح داده شده است ثبت می‌کنیم. در این مرحله کلاس مربوطه در QML登جیستر و شناسایی می‌شود.

قرار است با استفاده از مثال Google Api Map که به صورت زیر آمده است آن را سفارشی سازی کنیم:

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
        /* Always set the map height explicitly to define the size of the div
         * element that contains the map. */
        #map {
            height: 100%;
        }
        /* Optional: Makes the sample page fill the window. */
        html, body {
            height: 100%;
            margin: 0;
            padding: 0;
        }
    </style>
</head>
<body>
```

```

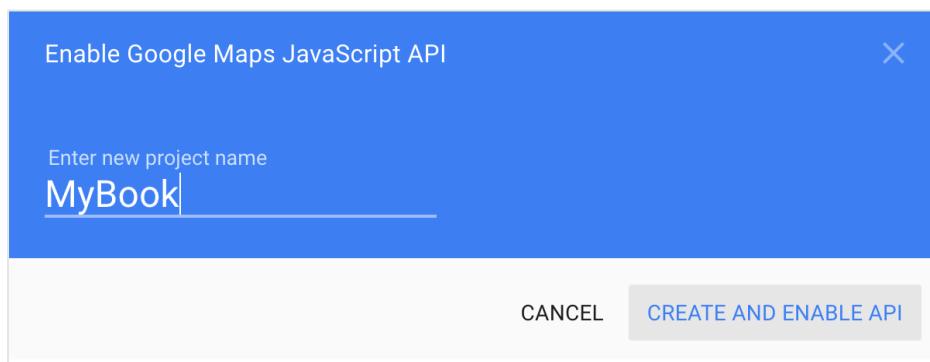
<div id="map"></div>
<script>
  var map;
  function initMap() {
    map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: -34.397, lng: 150.644},
      zoom: 8
    });
  }
</script>
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
  async defer></script>
</body>
</html>

```

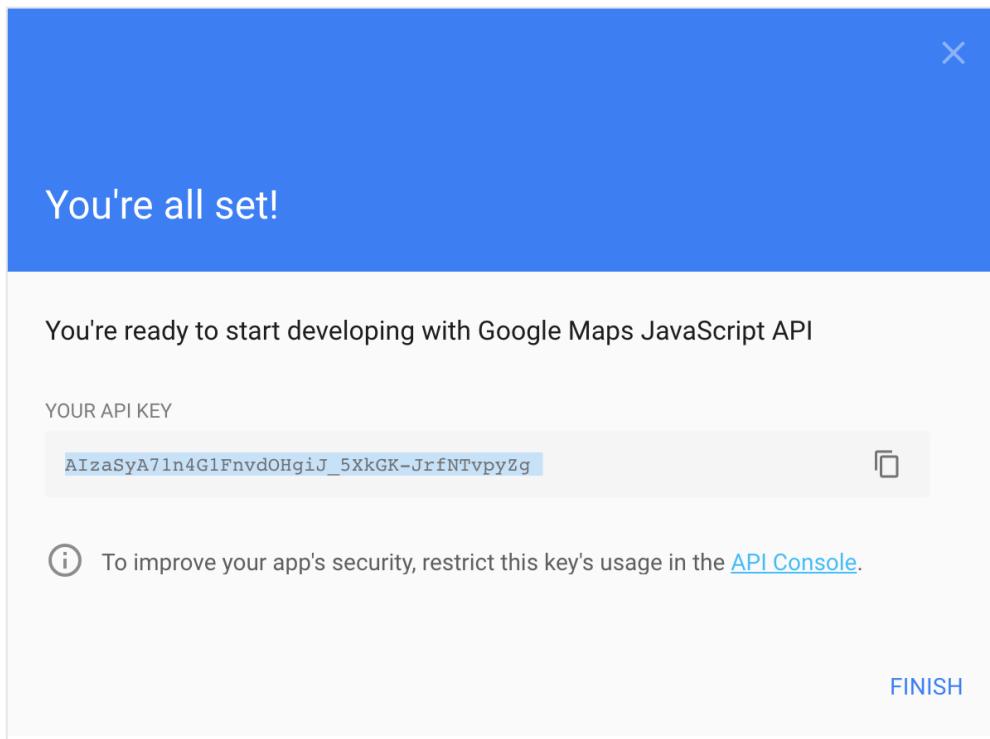
همانطور که مشخص است کد مربوطه مثالی از سند HTML است که بخشی از مستندات گوگل برای راهنمایی شما در رابطه با نحوه استفاده از سرویس Google Map به روش Web API است. در این مثال متغیرهای `lat`, `lng`, `zoom` مواردی هستند که ما به آنها نیاز خواهیم داشت سپس برای سفارشی سازی جهت استفاده از این سرویس متغیری باید برای `YOUR_API_KEY` تعریف شود. این گزینه برای دسترسی به سرویس گوگل نیاز است کافی است وارد کنسول گوگل شده و برای ساخت آن به روش زیر اقدام کنید. جهت دریافت کلید به لینک فوق مراجعه کنید :

<https://developers.google.com/maps/documentation/javascript/get-api-key>

بعد از ورود جهت ساخت شناسه توسعه برای محصول خود تصویر زیر را مشاهده خواهید کرد، نام پروژه خود را وارد کرده و گزینه `Create and enable api` را بزنید.



کلید ساخته شده از نوع رشته است، بنابراین به فکر این خواهیم بود که برای این نوع `QString` را در کلاس C++ تعریف کرده باشیم.



در ادامه، یک سند از نوع QML ساخته و نام آن را مشخص کنید. سپس کد زیر در درون سند QML نوشته خواهد شد:

```
import QtQuick 2.7
import QtQuick.Controls 2.0
import QtWebEngine 1.3
import QtQuick.Layouts 1.3
import Interface.Genyleap 1.0

Item {
    id: item
    width: 960
    height: 640

    WebEngineView { id:webView; anchors.fill: parent;

        onLoadProgressChanged: {

            if (webView.loadProgress === 100) {
                animation.visible = false
                progressContainer.visible = false
            }

            else {
                animation.visible = true
                progressContainer.visible = true
            }
        }
    }

    Map { id:map; }

    Rectangle {
```

```

        id: rectangle1
        y: 312
        height: 168
        color: "#ffffff"
        anchors.bottom: parent.bottom
        anchors.bottomMargin: 0
        anchors.left: parent.left
        anchors.leftMargin: 0
        anchors.right: parent.right
        anchors.rightMargin: 0

    GroupBox {
        id: groupBox1
        x: 21
        y: 15
        width: 888
        height: 138
        anchors.horizontalCenterOffset: 0
        anchors.bottom: parent.bottom
        anchors.bottomMargin: 15
        anchors.horizontalCenter: parent.horizontalCenter
        title: qsTr("Map mode")

        Button {
            id: button1
            x: 760
            y: 52
            text: qsTr("Day")
            onClicked: {

                webView.loadHtml('<!DOCTYPE html>
<html>
<head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
        html, body {
            height: 100%;
            margin: 0;
            padding: 0;
        }
        #map {
            height: 100%;
        }
    </style>
</head>
<body>
    <div id="map"></div>
    <script>
        var map;
        function initMap() {
            map = new google.maps.Map(document.getElementById('map'), {
                center: {lat: ' + map.lat(cx.text) + ', lng: ' + map.lng(cy.text) + '},
                zoom: ' + map.zoom(zoom.currentIndex) + '
            });
        }
    </script>
    <script src="https://maps.googleapis.com/maps/api/js?key=' + map.key(apikey.text)
+ '&callback=initMap"
        async defer></script>
</body>
</html>')
            }
        }
    
```

```

        }

    Button {
        id: button2
        x: 760
        y: 5
        text: qsTr("Night")
        onClicked: {

webView.loadHtml ('<!DOCTYPE html>
<html>
<head>
<title>Simple styled maps</title>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no">
<meta charset="utf-8">
<style>
    html, body {
        height: 100%;
        margin: 0;
        padding: 0;
    }
    #map {
        height: 100%;
    }
</style>
</head>
<body>
    <div id="map"></div>
    <script>
        function initMap() {
            var customMapType = new google.maps.StyledMapType([
                {
                    stylers: [
                        {hue: '\#890000'},
                        {visibility: 'simplified'},
                        {gamma: 0.5},
                        {weight: 0.5}
                    ]
                },
                {
                    elementType: 'labels',
                    stylers: [{visibility: 'off'}]
                },
                {
                    elementType: 'water',
                    stylers: [{color: '\#890000'}]
                }
            ], {
                name: 'Custom Style'
            });
            var customMapTypeId = '\custom_style';

            var map = new google.maps.Map(document.getElementById('map'), {
                zoom: '+' + map.zoom(zoom.currentIndex) + '',
                center: {lat: ' + map.lat(cx.text) + ', lng: ' + map.lng(cy.text) + '},
                mapTypeControlOptions: {
                    mapTypeIds: [google.maps.MapTypeId.ROADMAP, customMapTypeId]
                }
            });

            map.mapTypes.set(customMapTypeId, customMapType);
            map.setMapTypeId(customMapTypeId);
        }
    </script>
<script async defer

```

```

    src="https://maps.googleapis.com/maps/api/js?key=' + map.key(apikey.text) +
'&callback=initMap">
</script>
</body>
</html>' )
}
}

TextField {
    id: apikey
    x: 114
    y: -1
    width: 325
    height: 40
    text: qsTr("")
    placeholderText: "Enter your Google Developer api key!"
}

Label {
    id: label1
    x: 21
    y: 13
    text: qsTr("Enter you api key:")
}

Label {
    id: label2
    x: 41
    y: 59
    text: qsTr("Coordinate X:")
}

TextField {
    id: cx
    x: 114
    y: 45
    width: 123
    height: 40
    text: qsTr("33.3710522")
}

Label {
    id: label3
    x: 243
    y: 59
    text: qsTr("Coordinate Y:")
}

TextField {
    id: cy
    x: 316
    y: 45
    width: 123
    height: 40
    text: qsTr("50.5767876")
}

Label {
    id: label4
    x: 452
    y: 13
    text: qsTr("Zoom:")
}

ComboBox {
    id: zoom
    x: 497
}

```

```

        y: -1
        model: [ "1", "2", "3", "4", "5", "6", "7", "8" ]
        currentIndex: 6
    }
}

Rectangle {
    id:progressContainer
    anchors.centerIn: parent
    anchors.fill: parent
    visible: false

    AnimatedImage { id: animation; x: 560; y: 320; width: 64; height: 64;
        visible: false; currentFrame: 60; source: "qrc:/loading.gif"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
    }
}

}

```

با توجه با کد فوق ترکیب کدهای HTML, JS, QML و توابع همراه با کلاس‌های C++ مشخص است. مستند HTML مربوط به وب سرویس گوگل جهت نقشه بین تگ‌های دابل کوتیشن ("") قرار گرفته و سپس برابر با تابع loadHtml در WebView شده است. دقت کنید که برای ارسال توابع رجیستر شده در QML بین کدهای HTML کافی است از قوانین توسعه وب پیروی کنید. بین تگ‌های "" تنها توسط تگ ' و عملگر + می‌توانید مقدار دهی در درون HTML را انجام دهید. سپس بارگذاری نتیجه را به WebView می‌سپاریم.

برای اینکه در زمان بارگذاری صفحه تحت یک جلوه بصری شامل یک Loading تصویری را نمایش دهد ابتدا توسط یک AnimatedImage شناسه animation فایل با پسوند gif. خود را به آن اختصاص می‌دهیم. به صورت زیر:

```

AnimatedImage { id: animation; x: 560; y: 320; width: 64; height: 64;
    visible: false; currentFrame: 60; source: "qrc:/loading.gif"
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.verticalCenter: parent.verticalCenter
}

```

حال نیاز است کدی را پیاده سازی کنیم که در زمان بارگذاری محتوا تحت WebEngineView انیمیشنی نمایان شود برای این کار باید از خاصیت رویداد onLoadProgressChanged در WebEngineView استفاده شود. بنابراین کد آن به صورت زیر پیاده سازی شده است که تحت دستور شرطی در صورتی که محتوا در حال بارگذاری باشد تصویر مربوطه نمایش داده خواهد شد.

```

WebEngineView { id:webView; anchors.fill: parent;

onLoadProgressChanged: {

    if (webView.loadProgress === 100) {
        animation.visible = false
        progressContainer.visible = false
    }
    else {
        animation.visible = true
        progressContainer.visible = true
    }
}

```

در صورتی که مقدار webView در loadProgress با ۱۰۰ برابر باشد به معنی این است که بارگذاری کامل شده و در این صورت فایل مربوط به آنیمیشن مخفی خواهد شد. در غیر این صورت تا قبل از به اتمام رسیدن بارگذاری آنیمیشن نمایان خواهد بود.

نیاز است در زمان کلیک بر روی دکمه مقادیر ورودی از طرف QML دریافت سپس توسط C++ پردازش و مجدداً توسط HTML نمایش داده شود. جهت اینکار نوع TextField یا ComboBox را با شناسه‌های cx برای محور x ها و cy برای محور y ها در نظر می‌گیریم. همچنین نوع zoom را با شناسه zoom برای مشخص کردن مقدار zoom کننده در نقشه تعریف خواهیم کرد.

برای ارسال مقادیر توسط این کنترل‌ها کافی است مقادیر را برای متغیرهای موجود در html تحت JS تعریف کنیم:

```
<script>
var map;
function initMap() {
    map = new google.maps.Map(document.getElementById('map'), {
        center: {lat: ' + map.lat(cx.text) + ', lng: ' + map.lng(cy.text) + '},
        zoom: ' + map.zoom(zoom.currentIndex) +
    });
}
</script>
```

کد مربوطه با تعریف تابع‌های cx.text و cy.text این را فراهم می‌سازد که مقادیر lat و lng برای با توابع از پیش تعریف شده C++ قرار گیرند در واقع هر مقداری که دریافت شود تحت این توابع باز نگری خواهد شد.

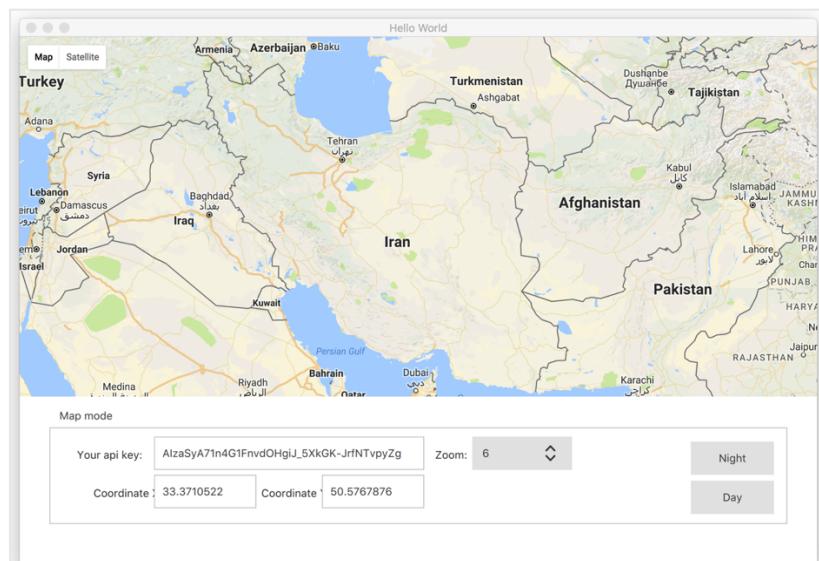
```
zoom: ' + map.zoom(zoom.currentIndex) + '
```

با توجه به اینکه zoom در این سرویس نوع عددی را به صورت شاخص دریافت می‌کند لازم است از ویژگی currentIndex مرتبط با استفاده شود تا مقادیر شاخص را برای این متغیر ارسال کند که مقدار دریافتی آن توسط تابع zoom() در C++ دریافت می‌شود.ComboBox

همچنین نوع TextField ای را برای دریافت apikey تعریف کرده‌ایم که با ارسال مقدار شناسه توسعه‌دهنده در گوگل امکان استفاده از آن را فراهم می‌کند. برای ارسال این مقدار کافی است کد JavaScript را با تابع C++ ترکیب کنیم.

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key='
+ map.key(apikey.text) + '&callback=initMap">
```

با دریافت مقدار رشته‌ای در کنترل **apikey** توسط متدهای text key از کلاس Map در C++ ارسال و مقدار آن را محاسبه خواهیم کرد. خروجی طرح ذکر شده به صورت زیر خواهد بود:



نکته: مازول Qt Web Engine با مازول Qt Web Assembly کاملاً متفاوت است، تفاوت مازول وب اسembly در این است که خود به عنوان یک پلتفرم در کیوت پشتیبانی می‌شود و در نهایت برنامه‌های تولید شده در محیط مرورگرهایی مانند کروم، موزیلا، اپرا، سافاری و غیره اجرا شوند.

## چند رسانه‌ای در کیوت

یکی از امکانات جالب کیوت پشتیبانی از محتوای چند رسانه‌ای به صورت کامل است، این مازول مجموعه‌ای از ویژگی‌های غنی را در بر دارد که اجازه می‌دهد تا دستگاه‌های مختلفی مانند دوربین، پخش ویدیو، پخش صوت و رادیو را طراحی کنید.

### ویژگی‌های این مازول

- دسترسی صوتی خام به درگاه‌های ورودی و خروجی بر روی دستگاه
- اجرای جلوه‌های صوتی با تاخیر کمتر
- پخش فایل‌های رسانه‌ای مانند فایل‌های فشرده شده ویدیو یا صوت
- ضبط صدا و فشرده سازی آن
- امکان گوش دادن به ایستگاه‌های رادیویی
- استفاده از یک دوربین، شامل منظره یاب، ضبط کننده تصویر و ضبط کننده ویدیو
- پخش صوت ۳ بعدی با استفاده از Qt Audio Engine
- رمزگشایی فایل‌های رسانه‌ای صوتی و ارسال در حافظه برای پردازش
- دسترسی به فریم‌های ویدیو و بافرهای صوتی که پخش یا ضبط می‌شوند

C++	QML	Use case
QSoundEffect		پخش جلوه صوتی
QAudioOutput		پخش صوت با تاخیر زمان کمتر
QMediaPlayer	Audio, MediaPlayer	پخش فایل‌های رمزگشایی شده مانند AAC و Mp3 ...
QAudioInput		دسترسی به داده‌های ورودی خام صوتی
QAudioRecorder		ضبط داده‌های صوتی رمزگشایی شده
QAudioDeviceInfo		کشف کننده دستگاه‌های داده‌های خام صوتی
QMediaPlayer, QAbstractVideoSurface, QVideoFrame	MediaPlayer, VideoOutput	پردازش کننده ویدیو

C++	نوع QML	Use case
QMediaPlayer, QVideoWidget, QGraphicsVideoItem	MediaPlayer, VideoOutput, Video	پخش کننده ویدیو
QMediaPlayer, QAbstractVideoSurface, QVideoFrame	MediaPlayer, VideoOutput	پردازش کننده ویدیو
QRadioTuner, QRadioData	Radio, RadioData	گوش دادن به رادیو
QCamera, QVideoWidget, QGraphicsVideoItem	Camera, VideoOutput	دسترسی به دوربین
QCamera, QAbstractVideoSurface, QVideoFrame	Camera, VideoOutput	پردازش تصویربرداری
QCamera, QCamerImageCapture	Camera	ضبط تصاویر
QCamera, QMediaRecorder	Camera	ظیبط ویدیو
AudioEngine, Sound	Audio Engine	منابع صوت سه بعدی

## چند رسانه‌ای در کیوت (صوت)

ماژول Qt Multimedia طیف وسیعی از کلاس‌های صوتی را فراهم می‌کند، همچنین سطح بالا و پایین ورودی و خروجی برای صدا و پردازش بر روی آن را فراهم کرده و برای استفاده از QML در این مورد انواع موجود Qt Audio Engine برای QML پیشنهاد می‌شود. به طور کلی پخش‌های رسانه‌ای یا صوتی به راحتی صورت نمی‌گیرد، صوت‌های غیر فشرده شده را می‌توانیم از کلاس‌های QMediaPlayer و یا انواع QML ای مانند Audio و MediaPlayer استفاده و پخش کنیم.

کلاس QMediaPlayer و انواع QML قادر به پخش ویدیو نیز هستند، پشتیبانی از قالب‌های صوتی فشرده با توجه به سیستم‌عامل قابل پشتیبانی است که در صورت نصب بودن پلاگین‌های مورد نیاز صورت می‌گیرد.

جهت استفاده از این ماژول آن را به پروژه اضافه می‌کنیم که در فایل pro. کد زیر را خواهیم داشت:

```
QT += multimedia
```

سورس سرآیند مربوط به کلاس را باز گذاری می‌کنیم:

```
#include <QMediaPlayer>
```

برای مثال کد ساده‌ای که برای پخش فایل صوتی در C++ خواهیم داشت به صورت زیر خواهد بود:

```
QMediaPlayer *player = new QMediaPlayer;
Player->setMedia(QUrl::fromLocalFile("./MySoundFile.mp3"));
player->setVolume(50);
player->play();
player->deleteLater();
```

در کد فوق از کلاس QMediaPlayer نمونه سازی کرده و سپس با افزودن مسیر فایل از نوع QUrl و سپس تنظیم صدای آن بر روی مقدار ۵۰ با متده play() آن را پخش می‌کنیم. در نهایت توسط متده deleteLater() پاکسازی حافظه گرفته شده از آن صورت می‌گیرد.

حال فرض کنید نیاز است لیستی از فایل‌ها را در لیست ذخیره کرده و سپس آن‌ها را بر اساس شاخص (index) پخش کنیم. در این صورت به صورت زیر از QMediaPlaylist استفاده خواهیم کرد که قبل از آن سرآیند مربوطه را فراخوانی می‌کنیم:

```
#include <QMediaPlaylist>
```

حال کد مربوطه به صورت زیر خواهد بود:

```
QMediaPlayer *player = new QMediaPlayer;
QMediaPlaylist *playlist = new QMediaPlaylist(player);
playlist->addMedia(QUrl("./mySoundFile1.mp3"));
playlist->addMedia(QUrl("./mySoundFile2.mp3"));
playlist->setcurrentIndex(1);
player->play();
player->deleteLater();
```

روش دیگر در رابطه با QML توسط وارد کردن مازول در سند QML به صورت زیر امکان‌پذیر است:

```
import QtMultimedia 5.9
```

سپس بعد از آن کافی است نوع QML را با نام MediaPlayer را با نام source ایجاد و مشخصه آن را برابر با فایل مورد نظر قرار دهیم.

```
MediaPlayer {
    id: playMusic
    source: ./MySoundFile.mp3
}

Component.onCompleted: {
    playMusic.volume = 0.5;
    playMusic.play();
}
```

در نهایت مشخصه volume را در بازه ۰.۰ تا ۱.۰ تنظیم و سپس توسط play پخش خواهد شد.

## چند رسانه‌ای در کیوت (صوت - ضبط صدا در یک فایل)

در نظر داشته باشید که توسط این مازول قرار است ما صدا (صوت) ای را ضبط و در یک فایل ذخیره سازی کنیم. کلاس QAudioRecord این امکان را فراهم می‌کند تا صوت را از دستگاه ورودی دریافت و به صورت فشرده شده در فایل ذخیره سازی کنیم. ابتدا سرآیند مربوط به کلاس مربوطه را فراخوانی می‌کنیم:

```
#include <QAudioRecorder>
```

سپس کد آن به صورت زیر خواهد بود:

```
QAudioRecorder *audioRecorder = new QAudioRecorder();
QAudioEncoderSettings audioSettings;

audioSettings.setCodec("audio/ mp3");
audioSettings.setQuality(QMultimedia::HighQuality);
audioRecorder->setEncodingSettings(audioSettings);

audioRecorder->setOutputLocation(QUrl::fromLocalFile("./MySound.mp3"));
audioRecorder->record();
audioRecorder->deleteLater();
```

ابتدا یک نمونه از کلاس QAudioRecorder ساخته و سپس نمونه‌ای از نوع کلاس QAudioEncoderSettings می‌سازیم. با خاصیت setCodec نوع فرمت صوت قابل ضبط را مشخص و سپس مقدار کیفیت آن را تنظیم می‌کنیم. در نهایت تنظیمات انجام شده اعمال و توسط setOutputLocation مسیر ذخیره فایل را مشخص می‌کنیم. سپس با متدهای record عمل ضبط آغاز شده و در نهایت حافظه تخصیص یافته از بین می‌رود.

## چند رسانه‌ای در کیوت (پخش ویدیو)

کلاس Qt Multimedia هر دو سطح بالا و پایین کلاس‌ها C++ را برای پخش و دستکاری داده‌های ویدئویی فراهم می‌کند، همچنین انواعی از نوع QML برای پخش و کنترل موجود هستند. البته بعضی از این کلاس‌ها با هر دو کلاس دوربین و صوت مشترک هستند که می‌تواند مفید باشد. بنابراین برای مشاهده مثال در نوع C++ به روش زیر ابتدا سرآیند کلاس مربوطه را وارد خواهیم کرد.

```
#include <QMediaPlayer>
#include <QMediaPlaylist>
#include <QVideoWidget>
```

سپس کد مربوط به پخش ویدیو به صورت زیر خواهد بود:

```
QMediaPlayer *player = new QMediaPlayer();
QMediaPlaylist *playlist = new QMediaPlaylist(player);

QVideoWidget *videoWidget = new QVideoWidget();

playlist->addMedia(QUrl("./myFile1.mp4"));
playlist->addMedia(QUrl("./myFile2.mp4"));
playlist->setCurrentIndex(1);
player->setVideoOutput(videoWidget);
videoWidget->show();

player->play();
player->deleteLater();
```

در این بخش برای آزمایش از کلاس QVideoWidget جهت ایجاد پنجره مربوطه استفاده شده است. فایل‌های ویدیو به ترتیب در لیست رسانه‌ها قرار گرفته و با مشخص کردن شاخص یکی از آن‌ها و ارسال و نمایش فایل صورت گرفته است. در نهایت همانند قبل تابع play() عمل پخش و بعد از پخش حافظه تخصیص یافته را آزاد می‌کند. deleteLater()

ممکن است نیاز باشد این کار را در بخش Front-End نیز انجام دهیم که در این صورت برای نوع QML عنوان Video را انتخاب خواهیم کرد. کافی است ماثول QML را وارد سند QtMultimedia کنیم.

```
import QtMultimedia 5.9

Video {
    id: video
    width : 800
    height : 600
    source: "./MyVideo.mp4"
}

Component.onCompleted: { video.play(); }
```

کد مربوط به فراخوانی فایل و پخش به صورت زیر است:

کد فوق نشان می‌دهد فایل مورد نظر در ابعاد ۸۰۰ در ۶۰۰ پیکسل بعد از بارگذاری کامل صفحه پخش خواهد شد. برای توضیحات بهتر برنامه‌ای را طراحی خواهیم کرد که به صورت یک دستگاه پخش کننده رسانه دیجیتالی عمل کند.

برنامه شامل قابلیت‌های زیر خواهد بود:

۱. دریافت فایل و پخش فرمت‌های مختلف در ابعاد مشخص

۲. اعمال تغییرات بر روی پخش (توقف، برگشت به عقب، حرکت به جلو، افزایش تن صدا و ...)

۳. پشتیانی در QML و طراحی سریع

بعد از افزودن ماژول multimedia در فایل pro. QML پخش کننده رسانه به صورت زیر خواهد بود:

```
import QtQuick 2.9
import QtQuick.Window 2.2
import QtQuick.Layouts 1.3
import QtMultimedia 5.8
import QtQuick.Controls 2.2
import QtQuick.Dialogs 1.2

Window {
    width: 800
    height: 600
    visible: true

    FileDialog {
        id: fileDialog
        title: "Please choose a file"
        folder: shortcuts.home
        nameFilters: [ "Video files (*.mp4 *.mpeg *.mpeg2 *.flv)", "All files (*)" ]
        onAccepted: {
            video.source = fileDialog.fileUrl;
            video.play();
        }
    }

    ColumnLayout {
        width: parent.width
        height: parent.height

        Rectangle {
            id: rectangle
            color:"#000"
            anchors.fill: parent

            Video {
                id: video
                anchors.fill: parent
            }
        }

        Rectangle {
            width: parent.width
            height: 64
            color:"#f1f1f1"
            anchors.bottom: parent.bottom

            RowLayout {
                spacing: 10
```

```
anchors.centerIn: parent

Button {
    text: "Select File"
    onClicked: {
        fileDialog.visible = true
    }
}

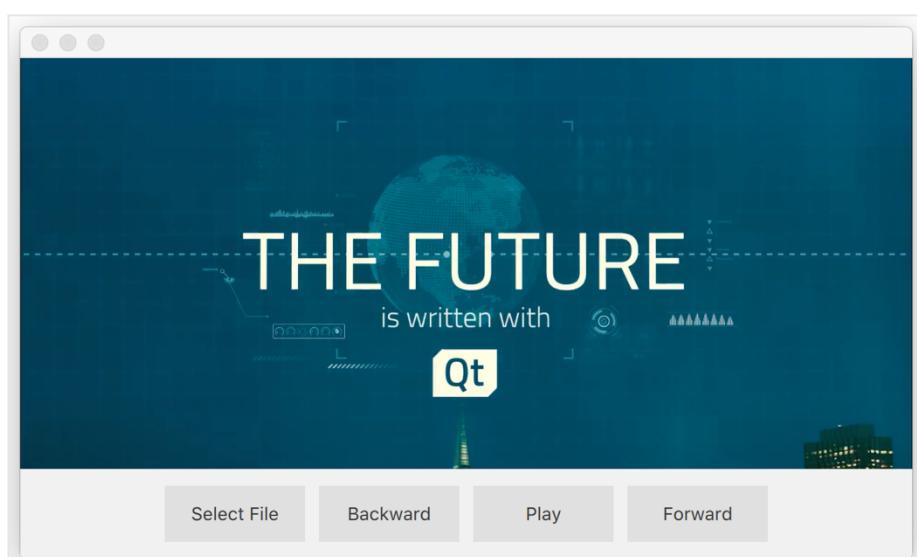
Button {
    text: "Backward";
    onPressed: {
        video.seek(video.position - 5000)
    }
}

Button {
    id:playButton
    text: "Play";
    onPressed: {
        video.playbackState == MediaPlayer.PlayingState
            ? video.pause() : video.play()
    }
}

Button {
    text: "Forward";
    onPressed: {
        video.seek(video.position + 5000)
    }
}

}
}
```

در این کد از مازول QtMultimedia و برخی از مازول‌هایی که قبلاً در مورد آن‌ها توضیح داده شده است استفاده می‌شود که بر اساس آن یک بخش، کنندهٔ سیار، ساده‌را در مدار MVP بساده سازی، کرده‌ام.



## چند رسانه‌ای در کیوت (کار با دوربین)

رابطه‌ای برنامه‌نویسی در QtMultimedia تعدادی از کلاس‌های مربوط به دوربین را فراهم می‌کند، بنابراین شما می‌توانید به تصاویر و ویدیوها از طریق دوربین‌ها و یا وب کم‌ها بر روی دستگاه‌های کامپیوتری و موبایل‌ها دسترسی داشته باشید. رابطه‌ای برنامه‌نویسی برای این قابلیت در هر دو روش C++ و QML موجود هستند.

برای دسترسی به اطلاعات سخت افزاری در رابطه با دوربین موجود بر روی دستگاه کلاس QCameraInfo مورد نیاز خواهد بود. همچنین برای دسترسی به دوربین و ارسال دستور نمایش و استفاده از آن از کلاس QCamera در C++ استفاده خواهیم کرد.

```
#include <QCameraInfo>

bool checkCameraAvailability()
{
    if (QCameraInfo::availableCameras().count() > 0)
        return true;
    else
        return false;
}
```

تابع checkCameraAvailability با دستور شرطی ارائه شده بررسی می‌کند و در صورتی که تابع availableCameras موجود در کلاس QCameraInfo مقدار بزرگتر از ۰ به معنی موجود و فعال بودن دوربین بر روی دستگاه را برگشت خواهد داد و مقدار آن برابر با true خواهد بود.

روش فوق در نوع QML به صورت زیر است:

```
property bool isCameraAvailable: QtMultimedia.availableCameras.length > 0
Component.onCompleted: console.log(isCameraAvailable)
```

برفرض اینکه دستگاه مربوطه دارای چند دوربین باشد در این صورت روشی برای تشخیص و انتخاب دوربین مورد نظر وجود دارد که در C++ به روش زیر بررسی لیستی از شناسه‌های انحصاری دوربین‌های موجود اعمال خواهد شد:

```
QList<QCameraInfo> cameras = QCameraInfo::availableCameras();
foreach (const QCameraInfo &cameraInfo, cameras) {
    qDebug() << cameraInfo.deviceName();
}
```

تابع deviceName() نام یا شناسه چند رقمی مناسب با دستگاه را باز می‌گرداند. اگر نیاز باشد بر اساس نام رشته آن دریافت شود کافی است از تابع description() استفاده شود تا مقدار بازگشتی آن برابر با نام دوربین شما باشد. در این صورت مقداری را ارسال خواهد که شامل نام سنسور دوربین بر روی دستگاه است به عنوان مثال : FaceTime HD Camera اگر انتخاب دوربین بین چند گزینه نیاز باشد به روش زیر عمل خواهیم کرد.

```
QList<QCameraInfo> cameras = QCameraInfo::availableCameras();
foreach (const QCameraInfo &cameraInfo, cameras) {
    if (cameraInfo.description() == "FaceTime HD Camera")
        qDebug() << "This is my selected camera!";
}
```

روش QML ای این ویژگی به صورت زیر است:

```
Camera { deviceId: QtMultimedia.availableCameras[0].deviceId }
```

جالب است بدانید روش بسیار جذابی در سوئیچ بین دوربین‌های عقب یا جلو وجود دارد که کیوت این کار را به راحتی انجام می‌دهد. در روش C++ به راحتی می‌توان بین دوربین‌ها سوئیچ کرد.

```
QCamera *camera = new QCamera( QCamera::FrontFace );
QCamera *camera = new QCamera( QCamera::BackFace );
```

در QML نیز به صورت زیر خواهد بود:

```
Camera { position: Camera.FrontFace }
Camera { position: Camera.BackFace }
```

برای نمایش دوربین به روش C++ کد زیر را خواهیم داشت:

```
QCamera* camera = new QCamera;
QCameraViewfinder* viewfinder = new QCameraViewfinder();
viewfinder->show();
camera->setViewfinder(viewfinder);
camera->start();
```

با ساخت نمونه از QCamera و نمونه‌ای از نمایه مخصوص جهت دوربین با نام QCameraViewerfinder در ادامه با نمایش قابل نمایه و در نهایت ارسال مقدار قابل بر روی دوربین و با استفاده از تابع start() عمل فعال‌سازی دوربین صورت می‌گیرد که نتیجه آن در قاب از پیش تعریف شده قابل نمایش است.

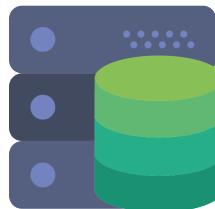
و البته در روش QML به صورت زیر خواهد بود:

```
VideoOutput {
    source: camera
    anchors.fill: parent
    Camera {
        id: camera
    }
}
```

## پیکربندی و معرفی کار با بانک اطلاعاتی (دیتابیس)

ماژول QSql یک ماژول ضروری است، این ماژول پشتیبانی از پایگاه‌های داده را فراهم می‌کند که در سه لایه جدا از هم قرار دارند:

- لایه درایور (راه انداز)
- لایه رابطه‌ای برنامه‌نویسی SQL
- لایه رابط کاربری



قبل از استفاده از SQL در کیوت باید دانش کافی را در رابطه با SQL داشته باشید، شما باید قادر به درک مطالب ساده بانک اطلاعاتی مانند دستورات کوئری SELECT, INSERT, UPDATE و DELETE وغیره... باشید. هرچند کلاس QSqlTableModel برای پیاده‌سازی رابط برای جستجو و ویرایش در دیتابیس را به گونه‌ای فراهم می‌کند که دانش زیادی در رابطه با SQL نیاز نباشد اما توصیه می‌شود که اطلاعات بیشتری را در زمینه کار با SQL کسب کنید.

برای استفاده از ماژول بانک اطلاعاتی در کیوت، کافی است آن را در فایل pro. خود اضافه کنید.

```
QT += sql
```

سپس برای فراخوانی و دسترسی در پروژه می‌توان از فایل سرآیند مرتبط با C++ به صورت زیر استفاده کرد.

```
#include <QtSql>
```

کلاس‌هایی برای کار با دیتابیس فراهم شده‌اند برای مثال کلاس **QSqlDatabase** ارتباط با دیتابیس را مدیریت می‌کند. برای مثال می‌توانیم تابعی برای اتصال به دیتابیس را بر پایه‌این کلاس بسازیم:

```
void MyDatabaseConnection() {  
  
    QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");  
    db.setHostName("localhost");  
    db.setDatabaseName("kambiz");  
    db.setUserName("root");  
    db.setPassword("");  
  
}
```

در حالت پیشفرض برقراری ارتباط با دیتابیس مقدور نخواهد بود و در صورتی که کد فوق که بر پایه درایور MySQL را اجرا کنید پیغام زیر را به عنوان عدم وجود درایور (راه‌انداز) مورد نظر دریافت خواهد کرد.

**QSqlDatabase:** QMYSQL driver not loaded

**QSqlDatabase:** available drivers: QSQLITE QMYSQL QMYSQL3 QODBC QODBC3 QPSQL QPSQL7

این پیغام به خاطر این است که درایور (راه‌انداز) دیتابیس بر روی کیوت نصب نشده است و باید بر اساس نیاز آن را همراه با منبع کد اصلی کیوت کامپایل (همگردانی) و راه‌اندازی نمایید که قبل از آموزش نصب به انواع درایورهایی اشاره می‌کنیم که در کیوت پشتیبانی می‌شوند.

در کل ماثول **QtSQL** برای راهاندازی برنامه‌های دارای پایگاه داده مورد استفاده قرار می‌گیرد برخی از درایورها به صورت پیشفرض با کیوت ارائه شده‌اند که برخی دیگر از آن‌ها را می‌توان به آن افزود.

در لیست زیر جدولی آمده است که شامل کیوت هستند اما به دلیل ناسازگاری با مجوز GPL همه آنها ارائه نشده‌اند بنابراین به صورت منبع باز همراه با کیوت ارائه نمی‌شوند.

نام درایور	توضیحات
QDB2	نسخه ۷.۱ و بالاتر از IBM-DB2
QIBASE	دیتابیس بورلند
QMYSQ	نوع مای اس کیو ال (MySQL)
QOCI	نوع اوراکل
QODBC	نوع دیتابیس مایکروسافتی (SQL Server)
QPSQL	نسخه ۷.۱ و بالاتر از PostgreSQL
QSQLITE2	نسخه ۲ SQLite
QSQLITE	نسخه ۳ SQLite

جهت حل این مسئله باید راهانداز مربوط به آن نصب شود، توجه داشته باشید که شما نیاز به فایل‌های سرآیند MySQL و کتابخانه‌های مشترک libmysqlclient.so در بستر لینوکس هستید. با توجه به توزیع ایستگاه یونیکس در لینوکس و مک با عنوان (**mysql-devel**) در قالب یک بسته قابل نصب هستند و همچنین در بستر ویندوز بسته‌هایی وجود دارد که می‌توانید آن‌ها را دریافت کنید.

#### پکیج توسعه جهت استفاده از درایور MySQL

(<https://dev.mysql.com/downloads/>)

<https://dev.mysql.com/downloads/mysql/>

قبل از هر چیز نیاز است بسته مربوطه را بر روی بستر مورد نظر خود نصب کنید، با توجه به اهمیت این بخش که به عنوان یکی از پرچالش‌ترین قسمت‌های کیوت برای کاربران تازه‌وارد محسوب می‌شود، سعی شده است تا به مراحل دقیق نصب و راهاندازی این موضوع پرداخته شود.

همانطور که می‌دانید در بستر ویندوز نسخه MinGW و MSVC برای کیوت موجود است، ما بر اساس نسخه MinGW درایور MySQL را راهاندازی خواهیم کرد. بنابراین، ابتدا بسته مربوطه را بر روی ویندوز در مسیر C:/MySQL/ استخراج کنید. سپس در این مسیر به دنبال پوشه‌های include و lib در MySQL باشید که دارای libmysql.lib نیز خواهد بود.

بعد از استخراج MySQL بر روی بستر ویندوز، در جستجوی پلتفرم خود برنامه (MinGW 5.3.0 32 bit) را نسبت به نسخه کیوت خودتان انتخاب کنید تا محیط کنسول مربوط به آن اجرا شود. توجه داشته باشید که از کنسول مختص مایکروسافت استفاده نکنید این بسیار مهم است. ابتدا به مسیر افزوده (پلاگین) مربوطه در محل نصب کیوت بروید تا به مسیر دستور دست افزونه دست پیدا کنید. برای مثال مسیر C:\Qt\Qt5.X.0-MinGW\5.X\Src\qtbase\src\plugins\sqldrivers\mysql که با دستور زیر به آن اشاره دارد:

```
cd C:\Qt\Qt5.X.0-MinGW\5.X\Src\qtbase\src\plugins\sqldrivers\mysql
```

سپس دستور بعدی جهت تنظیم راهانداز بر اساس MySQL نصب شده بر روی سیستم مورد نیاز خواهد بود:

```
qmake "INCLUDEPATH+=C:/MySQL/include" "LIBS+=C:/MySQL/MySQLServer<version>/lib/opt/libmysql.lib" mysql.pro
```

در صورتی که از کامپایلر مایکروسافت (MSVC) استفاده نمی‌کنید در این صورت از دستور nmake به جای make استفاده شود. فراموش نکنید که فایل **libmysql.dll** را از مسیر .../C:/MySQL/lib کپی کرده و در مسیر .../C:/Windows قرار دهید.

دستور زیر در ترمینال جهت اجرای ساخت پلاگین تحت پکیج MySQL در لینوکس و مکینتاش مورد استفاده قرار خواهد گرفت:

```
cd $QTDIR/qtbase/src/plugins/sqldrivers/mysql  
qmake "INCLUDEPATH+=/usr/local/mysql/include" "LIBS+=-L/usr/local/mysql/lib -lmysqlclient_r" mysql.pro  
make
```

بعد از اجرای دستور make، کافی است دستور install اجرا شود تا افزونهٔ مربوطه در مسیر مورد مناسب آن نصب شود. بعد از این کار فایلهای libmysqlclient.a را در لینوکس و libmysqlclient.so را در مسیر .../lib در مسیر .../usr/local/mysql قرار دهید. حال می‌توان تحت دستور زیر پیغام مناسب با وضعیت اتصال را دریافت کرد:

```
QSqlDatabase db (QSqlDatabase::addDatabase ("QMYSQL"));  
db.setHostName ("localhost");  
db.setDatabaseName ("mydatabase");  
db.setUserName ("root");  
db.setPassword ("");  
  
if(db.open ()) {  
    qDebug () << "Success!";  
} else {  
    qDebug () << db.lastError ().text ();  
}
```

کد فوق نشان می‌دهد در صورتی که اطلاعات وارد شده برای دیتابیس صحیح باشد و اتصال برقرار شود پیغام Success و در غیر این صورت کد پیغام خطاب برگشت داده خواهد شد.

نکته: استفاده از **QMYSQL** و همچنین **QMYSQ3** برای دسترسی به درایور MySQL امکان پذیر است.

روش فوق نمونه مثالی جهت ایجاد افزونهٔ راهانداز مربوطه برای بستر توسعه است، بنابراین در صورتی که نیاز باشد می‌توان از درایور **SQLite** نیز استفاده کرد که نیازی برای انجام این مراحل ندارد و برخلاف مدل‌های مرسوم بانک اطلاعاتی که به صورت **Client/Server** می‌باشند و نیاز به نصب و پیکربندی‌های خاص خوانش را دارند، SQLite تنها یک برنامه مدیریت بانک اطلاعاتی مستقل است که نیازی به هیچ گونه نصب و پیکربندی ندارد و مهمترین هدف از عرضه آن به کارگیری خاصیت ضمیمه شدن در سیستم‌های مختلف است. برای مثال استفاده در برنامه‌های موبایل و سیستم‌عامل‌های موبایل که با توجه به ماهیت آن‌ها است که ضرورت به کارگیری چنین برنامه‌ها، مدیریت بانک اطلاعاتی را دو چندان می‌کند. برای اینکار کافی است پروژه خود را بر پایه ساختار **SQLite** در کیوت ایجاد کنید.

کلاسی با نام **Database** به صورت زیر ایجاد کنید:

```

#ifndef DATABASE_H
#define DATABASE_H

#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QSqlRecord>

class Database
{
public:
    Database(const QString& path);
    bool addPerson(const QString& name);
    void selectPerson();
private:
    QSqlDatabase m_db;
};

#endif // DATABASE_H

```

در ادامه سورس کلاس به صورت زیر باز نویسی خواهد شد:

```

#include "database.h"
#include <QDebug>

Database::Database(const QString& path)
{
    m_db = QSqlDatabase::addDatabase("QSQLITE");
    m_db.setDatabaseName(path);

    if (!m_db.open())
    {
        qDebug() << "Error: connection with database fail";
    }
    else
    {
        qDebug() << "Database: connection ok";
    }
}

bool Database::addPerson(const QString& name)
{
    bool success = false;    QSqlQuery query;
    query.prepare("INSERT INTO people (name) VALUES (:name)");
    query.bindValue(":name", name);
    if(query.exec())
    {
        success = true;
    }
    else
    {
        qDebug() << "addPerson error: "
                  << query.lastError();
    }

    return success;
}

void Database::selectPerson ()
{
    QSqlQuery query("SELECT * FROM people");
    int idName = query.record().indexOf("name");

```

```

while (query.next())
{
    QString name = query.value(idName).toString();
    qDebug() << "Name = " << name;
}
}

```

در کلاس فوق پارامتر path با نوع `QString` جهت مشخص کردن مسیر فایل دیتابیس با پسوند db از نوع SQLite است. در ادامه تابع `void void addPerson` با نام `selectPerson` و تابع `addDatabase` در کلاس `QSqlDatabase` موجود است. در نظر داشته باشید برای مشخص سازی نوع درایور از تابع `addDatabase` پیش‌فرض `QSqlDatabase` استفاده می‌شود که مقدار آن با توجه به نوع مثال `QSQLITE` تعیین شده است. در ادامه متدهای `setDatabaseName` مسیر و نام مربوط به فایل را دریافت و در آخر با استفاده از شرط مربوطه بررسی می‌شود که فایل مربوط به دیتابیس موجود است یا خیر.

تابع `addPerson` همراه با پارامتر ورودی `name` از نوع `QString` با استفاده از کلاس  `QSqlQuery` مقدار دهی شده است که از استاندارد SQL پشتیبانی می‌کند. همچنین تابع `selectPerson` با استفاده از کوئری مربوطه اطلاعات مربوط به رکورد موجود را دریافت می‌کند.

جهت اجراء دستورات ایجاد رکورد به صورت زیر عمل خواهیم کرد:

```
Database *db = new Database(".../.../MyProject/MyDatabase.db");
```

مسیر مربوطه و نام فایل را همراه با پسوند برای کلاس تعریف و سپس از تابع `addPerson` جهت افزودن نام شخص به صورت زیر استفاده می‌کنیم.

```
db->addPerson ("Kambiz");
```

با اجرای دستور فوق نام مربوطه در دیتابیس ذخیره می‌شود. حال جهت دریافت آن کافی است تابع `selectPerson` را اجرا کنیم:

```
db->selectPerson ();
```

خروجی دستورات فوق در کنسول محیط توسعه به صورت زیر خواهد بود:

```
Database: connection ok
Name = "Kambiz"
```

مسیر مربوطه و نام فایل را همراه با پسوند برای کلاس تعریف و سپس از تابع `addPerson` جهت افزودن نام شخص به صورت زیر استفاده می‌کنیم.

## کار با بانک اطلاعاتی و ارتباط آن بین C++ و QML

هرچند در QML می‌توان از دستورات SQL نیز استفاده کرد اما اصول کار بر این است که در بک‌اِнд برنامه از دستورات قدرتمند توسط C++ استفاده شود که در این میان کلاس  `QSqlQueryModel` امکان این را فراهم می‌کند تا مدل‌های پیاده سازی شده را برای QML ارسال کنیم. ساده‌ترین روش ممکن برای آشنایی با این کلاس به صورت زیر خواهد بود:

```
QSqlQueryModel *model = new QSqlQueryModel;
model->setQuery("SELECT firstname, lastname FROM users");
QQmlApplicationEngine engine;
engine.rootContext()->setContextProperty("myDataModel", model);
```

در کد ذکر شده ابتدا مدلی از نوع `QSqlQueryModel` ایجاد و سپس توسط متدهای `setQuery` و `setContextProperty` دستور SQL مربوطه را اجرا می‌کنیم. سپس توسط موتور مربوطه خاصیت مدل را با نام مستعار `myDataModel` برای QML ارسال می‌کنیم.

```
ListView {
    width: 200; height: 200
    model: myDataModel
    delegate: Row {
```

```

    Rectangle {
        width: 100; height: 40
        Text {
            anchors.fill: parent
            text: display
        }
    }
}

```

نوع **ListView** با دریافت مدل ارسال شده از طرف C++ با نام myDataModel توسط کلید ثابت display دادهای ارسال شده را در قالب رشته نمایان می‌سازد. نامهای از قبل تعریف شده به صورت کلید whatsThis و display, decoration, edit, toolTip, statusTip هستند. که به ترتیب display جهت نمایش داده در قالب رشته، edit جهت نمایش داده با قابلیت ویرایش مناسب، decoration جهت نمایش داده در قالب آیکون بر اساس کلاس‌های QIcon، QPixmap یا QColor و همچنین toolTip جهت نمایش آیتم به صورت راهنمای، statusTip برای نمایش داده در قالب نوار وضعیت و whatsThis جهت نمایش آیتم در قالب سفارشی What's This می‌باشند.

جهت سفارشی سازی و نحوه استفاده از مدل‌های داده‌ای بر پایه SQL و ارسال آن بر روی QML باید کلاس‌های آن را تحت C++ بر اساس **QSqlQueryModel** پیاده سازی کنیم. بنابراین کلاس زیر نمونه‌ای از آن خواهد بود:

```

#ifndef LISTMODEL_H
#define LISTMODEL_H

#include <QObject>
#include <QSqlQueryModel>

class ListModel : public QSqlQueryModel
{
    Q_OBJECT
public:
    enum Roles {
        IdRole = Qt::UserRole + 1,           // Id
        FirstNameRole,                     // First Name
        LastNameRole,                      // Last Name
        MobileRole                         // Mobile Number
    };
    explicit ListModel(QObject *parent = 0);
    QVariant data(const QModelIndex & index, int role = Qt::DisplayRole) const;
protected:
    QHash<int, QByteArray> roleNames() const;
signals:
public slots:
    void updateModel();
    int getId(int row);
};

#endif // LISTMODEL_H

```

ابتدا کلاسی بر پایه ListModel با نام QSqlQueryModel ساخته و سپس Role های مرتبط با آن را در قالب شمارشی بر اساس کلاس ارائه دهنده در قالب شمارنده تعریف می‌کنیم. آیتم‌ها شامل نام، نام خانوادگی و شماره موبایل خواهند بود. در ادامه با ضمیمی سازی کلاس

در سازنده و تعریف تابع `data` و سیگنال‌های مورد نظر بدنه کلاس را ایجاد می‌کنیم. سپس برای باز تعریف توابع به طور کامل محتوای فایل `listmodel.cpp`. به صورت زیر خواهد بود:

```
#include "listmodel.h"
#include "database.h"

ListModel::ListModel(QObject *parent) :
    QSqlQueryModel(parent)
{
    this->updateModel();
}

QVariant ListModel::data(const QModelIndex & index, int role) const {
    int columnId = role - Qt::UserRole - 1;
    QModelIndex modelIndex = this->index(index.row(), columnId);

    return QSqlQueryModel::data(modelIndex, Qt::DisplayRole);
}

QHash<int, QByteArray> ListModel::roleNames() const {
    QHash<int, QByteArray> roles;
    roles [IdRole]         = "id";
    roles [FirstNameRole] = "firstname";
    roles [LastNameRole]  = "lastname";
    roles [MobileRole]    = "mobile";
    return roles;
}

void ListModel::updateModel()
{
    this->setQuery("SELECT id, firstname, lastname, mobile FROM " TABLE);
}

int ListModel::getId(int row)
{
    return this->data(this->index(row, 0), IdRole).toInt();
}
```

کلاس دیتابیس جهت برقراری ارتباط با دیتابیس و اتصال به صورت زیر است:

```
#ifndef Database_H
#define Database_H

#include <QObject>
#include <QSql>
#include <QSqlQuery>
#include <QSqlError>
#include <QSqlDatabase>
#include <QFile>
#include <QDate>
#include <QDebug>

#define Database_HOSTNAME    "localhost"
#define Database_NAME        "mydatabase.db"

#define TABLE                 "MyTable"

class Database : public QObject
{
    Q_OBJECT
public:
```

```

explicit Database(QObject *parent = 0);
~Database();
void connectToDatabase();

private:
    QSqlDatabase     db;

private:
    bool openDatabase();
    bool restoreDatabase();
    void closeDatabase();
    bool createTable();

public slots:
    bool inserIntoTable(const QVariantList &data);           // Adding entries to the
    table
    bool inserIntoTable(const QString &fname, const QString &sname, const QString
&nik);
    bool removeRecord(const int id); // Removing records from the table on its id
};

#endif // Database_H

```

و فایل Database.cpp آن نیز دارای محتوای زیر خواهد بود:

```

#include "database.h"

Database::Database(QObject *parent) : QObject(parent)
{
}

Database::~Database()
{
}

void Database::connectToDatabase()
{
    if (!QFile("/Users/Compez/Documents/Projects/Book/" Database_NAME).exists()) {
        this->restoreDatabase();
    } else {
        this->openDatabase();
    }
}

bool Database::restoreDatabase()
{
    if (this->openDatabase()) {
        return (this->createTable()) ? true : false;
    } else {
        qDebug() << "Failed to restore the Database";
        return false;
    }
    return false;
}

bool Database::openDatabase()
{
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setHostName(Database_HOSTNAME);
    db.setDatabaseName("/Users/Compez/Documents/Projects/Book/" Database_NAME);
    if (db.open()) {
        return true;
    }
}

```

```

    } else {
        return false;
    }
}

void Database::closeDatabase()
{
    db.close();
}

bool Database::createTable()
{
    QSqlQuery query;
    if(!query.exec( "CREATE TABLE " TABLE " ("
                    "id INTEGER PRIMARY KEY AUTOINCREMENT, "
                    "firstname VARCHAR(255)      NOT NULL, "
                    "lastname VARCHAR(255)      NOT NULL, "
                    "mobile VARCHAR(255)      NOT NULL"
                    ")"
                )) {

        qDebug() << "Database: error of create " << TABLE;
        qDebug() << query.lastError().text();
        return false;
    } else {
        return true;
    }
    return false;
}

bool Database::inserIntoTable(const QVariantList &data)
{
    QSqlQuery query;
    query.prepare("INSERT INTO " TABLE " ( firstname, "
                  "lastname, "
                  "mobile ) "
                  "VALUES (:firstname, :lastname, :mobile)");

    query.bindValue(":firstname",           data[0].toString());
    query.bindValue(":lastname",            data[1].toString());
    query.bindValue(":mobile",             data[2].toString());

    if(!query.exec()){

        qDebug() << "error insert into " << TABLE;
        qDebug() << query.lastError().text();
        return false;
    } else {
        return true;
    }
    return false;
}

bool Database::inserIntoTable(const QString &fname, const QString &sname, const
QString &nik)
{
    QVariantList data;
    data.append(fname);
    data.append(sname);
    data.append(nik);

    if(inserIntoTable(data))
        return true;
    else
        return false;
}

```

```

}

bool Database::removeRecord(const int id)
{
    QSqlQuery query;

    query.prepare("DELETE FROM " TABLE " WHERE id= :ID ;");
    query.bindValue(":ID", id);

    if(!query.exec()) {
        qDebug() << "error delete row " << TABLE;
        qDebug() << query.lastError().text();
        return false;
    } else {
        return true;
    }
    return false;
}

```

کلاس‌های فوق نمونه مثالی است جهت برقراری ارتباط با دیتابیس عنوان شده است و جهت پیاده سازی و دسترسی عملیاتی بر روی QML به ابتدا آن را ثبت خواهیم کرد به صورت زیر:

```

#include <QApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext>

#include "database.h"
#include "listmodel.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QQmlApplicationEngine engine;

    Database database;
    database.connectToDatabase();

    ListModel *model = new ListModel();

    engine.rootContext()->setContextProperty("myModel", model);
    engine.rootContext()->setContextProperty("database", &database);

    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}

```

کلاس‌های فوق نمونه مثالی است جهت برقراری ارتباط با دیتابیس عنوان شده است و جهت پیاده سازی و دسترسی عملیاتی بر روی QML به ابتدا آن را ثبت خواهیم کرد به صورت زیر:

```

import QtQuick 2.5
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.3
import QtQuick.Dialogs 1.2

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    ColumnLayout {

```

```

    id: rowLayout
    anchors.top: parent.top
    anchors.left: parent.left
    anchors.right: parent.right
    anchors.margins: 5
    spacing: 10

    Text {text: qsTr("Firstname")}
    TextField {id: firstnameField}
    Text {text: qsTr("Lastname")}
    TextField {id: lastnameField}
    Text {text: qsTr("Mobile")}
    TextField {id: mobileField}

    Button {
        text: qsTr("Add Data")
        onClicked: {
            database.insertIntoTable(firstnameField.text, lastnameField.text,
mobileField.text)
            myModel.updateModel()
        }
    }

    Button {
        text: "Remove"
        onClicked: contextMenu.open();
    }

}

ListView {
    id: tableView
    anchors.top: rowLayout.bottom
    anchors.left: parent.left
    anchors.right: parent.right
    anchors.bottom: parent.bottom
    anchors.margins: 5
    anchors.topMargin: 30
    model: myModel
}

Row {
    id:rl
    x: 0
    y: -30
    Text { text: "Firstname"; font.bold: true; width: 120; }
    Text { text: "Lastname"; font.bold: true; width: 120; }
    Text { text: "Mobile"; font.bold: true; width: 120; }
    spacing: 10
}

delegate: RowLayout {
    spacing: 10
    MouseArea {
        id:mouseArea
        anchors.fill: parent
        hoverEnabled: true
    }

    Rectangle {
        id:rc;
        anchors.fill: parent;

```

```

        color: mouseArea.containsMouse ? "#ccc" : "#fff"
    }

Column { Text { text: firstname; width: 120; } }
Column { Text { text: lastname; width: 120; } }
Column { Text { text: mobile; width: 120; } }

}

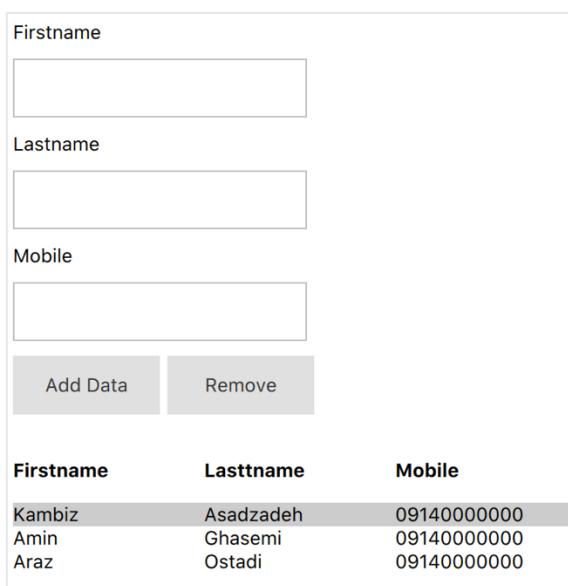
Menu {
    id: contextMenu
    MenuItem {
        text: qsTr("Remove")
        onTriggered: {
            dialogDelete.open()
        }
    }
}

MessageDialog {
    id: dialogDelete
    title: qsTr("Remove record")
    text: qsTr("Confirm the deletion of log entries")
    icon: StandardIcon.Warning
    standardButtons: StandardButton.Ok | StandardButton.Cancel

    onAccepted: {
        database.removeRecord(myModel.getId(tableView.currentRow))
        myModel.updateModel();
    }
}
}

```

همانطور که در فصل‌های قبلی به انواع QML و کلاس‌های C++ و نحوه برقراری ارتباط بین آن‌ها اشاره شد در کد مربوطه امکان ایجاد دیتابیس و برقراری ارتباط بین بک‌اند و فرانت اند صورت گرفته است که کافی است در قالب یک پروژه کدهای موجود را ارسال و اجرا نمایید.



## معرفی و کار با XML

بر اساس تعاریف رسمی این فناوری (اکس‌ام‌ال) - XML یا زبان نشانه‌گذاری گسترش‌پذیر مخفف (eXtensible Markup Language) را باید بدون تردید یکی از بزرگ‌ترین و اساسی‌ترین گامهایی به حساب آورد که در مسیر حل مشکل اندازه‌پذیری در اینترنت مدرن برداشته شده است. اکس‌ام‌ال

ویرایشی از اس جی ام ال است که می‌کوشد فاصله بین سادگی اجتناب‌آور و قدرت اس جی ام ال پل بزند. در واقع اس جی ام ال زیر مجموعه‌ای از اس جی ام ال است که صرفاً برای استفاده با وب طراح شده است. پس از ایجاد اکس ام ال توسط کنسرسیوم وب جهانگیر (W3C) در سال ۱۹۹۶ (میلادی)، دست‌اندرکاران بسیاری از پروژه‌های محاسبات توزیع شده به استفاده گسترده از آن روی آوردند.

حال برای پیاده سازی و استفاده از این فناوری در Qt خواهیم پرداخت که در پروژه‌ها و محصولات استارت‌اپی پیشرفت‌ه کاربرد بسیار زیادی دارد و لازم است رابطه آن را درک و از فواید آن بهره‌مند شویم. بنابراین در کیوت کلاس‌های **QXmlStreamWriter** و **QXmlStreamReader** به ترتیب جهت تجزیه کردن (خواندن) اطلاعات و نوشتن و تولید ساختار استاندارد XML در فایل یا داده را فراهم می‌کنند.

قبل از هر چیز جهت دسترسی به این ماژول کافی است دستور زیر را ابتدا در فایل pro. اضافه کنید.

```
QT += xml
```

سپس برای استفاده از کلاس‌های مربوطه سرآیند مربوط به آن را فراخوانی خواهیم کرد:

```
#include <QtXml>
```

کلاس **QXmlStreamReader** امکان تجزیه کردن داده‌ها در قالب XML را به راحتی فراهم می‌سازد. بنابراین کد زیر مثالی است در رابطه با این موضوع که نحوه تجزیه کردن محتوای فایل XML را نشان می‌دهد:

```
QString key;
 QFile* file = new QFile("../test.xml");
 if(!file->open(QIODevice::ReadOnly | QIODevice::Text)){
     qDebug() << "Failed To Open Xml";
     return -1;
 }

QXmlStreamReader* xml = new QXmlStreamReader(file);

while(!xml->atEnd()){

    QXmlStreamReader::TokenType token = xml->readNext();
    if(token == QXmlStreamReader::StartDocument)

        if(xml->name() == "Person"){
            qDebug() << "Book Selected!";
        }

    if(xml->name() == "firstname"){
        key = xml->readElementText();
        qDebug() << "Firstname : " << key;
    }

    if(xml->name() == "lastname"){
        key = xml->readElementText();
        qDebug() << "Lastname : " << key;
    }
}
```

محتوای فایل به صورت زیر است:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<Person>
    <firstname>Kambiz</firstname>
    <lastname>Asadzadeh</lastname>
    <city>Urmia</city>
</Person>

```

ابتدا جهت دسترسی به فایل مربوطه از کلاس QFile استفاده شده است که جهت ارسال فایل برای کلاس QXmlStreamReader فراهم شده است که قبل از رسیدن به انتهای محتوا در فایل توسط متد atEnd بررسی می‌شود که آیا فایل تا انتها قابل خواندن بوده است یا خیر. در صورتی که مشکلی موجود باشد مقدار false را بازگشت خواهد داد که در این کد در ادامه تحت دستور شرطی ساده‌ای توسط if مشخص می‌کنیم که نام اولین کلید به عنوان Person را بررسی کرده و پیغام مربوطه را چاپ خواهد کرد. این عمل به صورت شرطی برای کلیدهای دیگر مانند firstname و lastname صورت می‌گیرد که نتیجه هریک دسترسی و چاپ مقادیر موجود در آن‌ها است که محتوای کلید مربوطه توسط متد readElementText دریافت و برای ورودی چاپ توسط std::cout و یا qDebug() ارسال می‌شود.

همچنین جهت نوشتن در فایل و قالب XML کد زیر را خواهیم داشت:

```

// Create a document to write XML
QDomDocument document;

// Making the root element
QDomElement root = document.createElement("Person");

// Adding the root element to the document
document.appendChild(root);

// Adding more elements

QDomElement firstname = document.createElement("firstname");
firstname.setAttribute("firstname", "Kambiz");
root.appendChild(firstname);

QDomElement lastname = document.createElement("lastname");
lastname.setAttribute("lastname", "Asadzadeh");
root.appendChild(lastname);

QDomElement city = document.createElement("city");
city.setAttribute("city", "Urmia");
root.appendChild(city);

// Writing to a file
QFile file("../test.xml");
if(!file.open(QIODevice::WriteOnly | QIODevice::Text))
{
    qDebug() << "Open the file for writing failed";
}
else
{
    QTextStream stream(&file);
    stream << document.toString();
    file.close();
    qDebug() << "Writing is done";
}

```

بر خلاف تجزیه سند در قالب XML جهت ایجاد و ارسال اطلاعات در داخل XML کلاسی با نام QDomDocument وجود دارد که امکان ارائه سند در قالب XML را فراهم می‌سازد. سپس کلاس QDomElement به عنوان مشخص کننده عناصر در قالب XML را فراهم خواهد ساخت. کافی است از تابع createElement استفاده و مقدار عنصر مورد نظر را مشخص سازیم. در ادامه متده appendChild عنصر مورد نظر را به عنوان فرزند در زیر مجموعه بالاترین عنصر معین می‌سازد. در نهایت توسط متده setAttribute صفت عنصر و ارزش (مقدار) مورد نظر را ارسال می‌کنیم. در نهایت توسط کلاس QFile مسیر و فایل مربوطه مشخص شده و با مشخصه‌های آن محتوای مربوطه در فایل ذخیره می‌شود که خروجی آن به صورت زیر خواهد بود:

```
<Person>
  <firstname firstname="Kambiz"/>
  <lastname lastname="Asadzadeh"/>
  <city city="Urmia"/>
</Person>
```

معمولًاً نیاز می‌شود که در بالاترین سطح ممکن یعنی در فرانت‌اِند استفاده از این قابلیت فراهم شود که توسط QML می‌توان از آن بهره‌مند شد. تنها کافی است مازول مربوطه را در سند QML فراخوانی کنید.

```
import QtQuick.XmlListModel 2.0
```

کلاس فوق مشخصه‌هایی مانند name، isKey و query را در اختیار قرار می‌دهد که بر اساس آن سند زیر را تجزیه خواهیم کرد:

```
<Person>
  <firstname>Kambiz</firstname>
  <lastname>Asadzadeh</lastname>
  <city>Urmia</city>
</Person>
```

کد زیر جهت ایجاد مدل XML است که در آن مشخصه source برای دریافت مسیر سند و query جهت رشته پرس‌جو در سند مشخص شده است. XmlRole نیز جهت مشخص سازی مدل برای سند است که تحت گزینه‌های firstname، lastname و city تعریف شده است. توجه داشته باشید که name نام گزینه‌ها را مشخص می‌کند و query مشخصه و نوع داده تعیین می‌کند.

```
XmlListModel {
    id: xmlModel
    source: "../test.xml"
    query: "/Person"

    XmlRole { name: "firstname"; query: "firstname/string()" }
    XmlRole { name: "lastname"; query: "lastname/string()" }
    XmlRole { name: "city"; query: "city/string()" }
}

ListView {
    width: 180; height: 300
    model: xmlModel
    delegate: ColumnLayout {
        Text { text: "Firstname : " + firstname}
        Text { text: "Lastname : " + lastname}
        Text { text: "City : " + city}
    }
}
```

در نهایت تحت کنترل ListView و مشخصه مدلینگ داده‌ای در قالب یک Column محتوای XML را در QML نمایش می‌دهیم.

بر اساس تعاریف رسمی جی سان (JSON) مخفف JavaScript Object Notation نشانه‌گذاری شئ جاوااسکریپت، یک استاندارد باز متنی سبک برای انتقال داده‌ها است به گونه‌ای که برای انسان نیز خوانا باشد. جی سان از زبان اسکریپت‌نویسی جاوااسکریپت در نشان دادن ساختمان داده‌های ساده و آرایه‌های انجمنی مشتق شده است. با وجود ارتباط عمیقی که با جاوااسکریپت دارد، جی سان مستقل از زبان است و مفسرهاش تفریباً برای هر زبانی موجود هستند.

در کیوت برای استفاده از این فناوری تحت کلاس‌های C++ روش بسیار ساده‌ای وجود دارد که به طور کلی ساختار شش نوع داده پشتیبانی می‌شود که شامل bool, double, string, array, object و null است. در جی سان متغیرها می‌توانند شامل انواع مختلفی باشند برای مثال نوع bool یا string و غیره... که نسبت به هر یک باید توجه داشته باشیم در تجزیه آن‌ها از نوع‌های مرتبط استفاده کنیم.

```
{
    "FirstName": "Kambiz",
    "LastName": "Asadzadeh",
    "Mobile": 09140000000,
    "Email": [
        "kambiz.ceo@gmail.com",
        "info@genyleap.com"
    ]
}
```

قالب بالا شکلی از نوع رشته و عددی در سند JSON را نشان می‌دهد. برای کار با این فناوری کلاس QJsonDocument امکان این را فراهم می‌سازد که خواند و نوشتمن را در سند جی سان را در حالت رمزنگاری شده در استاندارد UTF-8 را در اختیار داشته باشیم.

با توجه به این که با فناوری JSON آشنا هستید جهت خواندن و نوشتمن این نوع سند مثالی آورده می‌شود که ابتدا باید سرآیند کلاس‌های زیر را فراخوانی کنیم که به ترتیب جهت دسترسی به نوع سند جی سان، آرایه‌ها و اشیاء مورد استفاده قرار خواهد گرفت.

```
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
```

در ادامه، کد زیر جهت خواندن (تجزیه) کردن سند JSON پیاده سازی شده است.

```
QString val;
 QFile file;
 file.setFileName("../test.json");
 file.open(QIODevice::ReadOnly | QIODevice::Text);
 val = file.readAll();
 file.close();

 QJsonDocument d = QJsonDocument::fromJson(val.toUtf8());
 QJsonObject sett2 = d.object();
 QJsonValue value = sett2.value(QString("appName"));
 qDebug() << value;
 QJsonObject item = value.toObject();
 QJsonValue subobj = item["description"];
 qDebug() << subobj.toString();

 QJsonArray test = item["imp"].toArray();
 qDebug() << test[2].toString();
```

در کد فوق برای دسترسی به شیء موجود در سند جیسان نوعی از `QJsonObject` را ایجاد کرده و سپس برای دسترسی به مقادیر آن از کلاس `QJsonValue` استفاده شده است. در نهایت با قرار دادن کلید یا شناسه مقادیر مورد نظر در پارامتر `value` ارزش هر یک را دریافت و توسط مجرای `qDebug()` چاپ می‌کنیم. فایل سند JSON دارای محتوای زیر است:

```
{
    "appDesc": {
        "description": "SomeDescription",
        "message": "SomeMessage"
    },
    "appName": {
        "description": "Home",
        "message": "Welcome",
        "imp": ["awesome", "best", "good"]
    }
}
```

خروجی کد مربوطه مقادیر `description` برابر با `Home` و `imp` برابر با `best` را دریافت خواهد کرد. جهت ساخت سند جیسان با محتوای مشابه زیر در ادامه کد مربوط به آن را مشاهده خواهید کرد:

```
[
    {
        "CPP-Ver": 17,
        "FirstName": "Kambiz",
        "LastName": "Asadzadeh",
        "Phone Numbers": [
            "+98 9140000000",
            "+98 4400000000"
        ]
    }
]
```

جهت ساخت محتوا در قالب جیسان اشیاء خود را بر اساس کلاس `QJsonObject` تعریف کرده و سپس برای درج اطلاعات در آن کافی است از متده استفاده شود که ابتدا کلید و یا شناسه انحصاری هریک را دریافت و سپس مقدار مربوطه توسط `QJsonValue` مشخص می‌شود که در این مثال از متده استفاده شده است تا تمامی انواع پشتیبانی شوند.

```
QJsonObject recordObject;
recordObject.insert("FirstName", QJsonValue::fromVariant("Kambiz"));
recordObject.insert("LastName", QJsonValue::fromVariant("Asadzadeh"));
recordObject.insert("CPP-Ver", QJsonValue::fromVariant(17));

QJsonArray phoneNumbersArray;
phoneNumbersArray.push_back("+98 9140000000");
phoneNumbersArray.push_back("+98 4400000000");
recordObject.insert("Phone Numbers", phoneNumbersArray);

QJsonArray recordsArray;
recordsArray.push_back(recordObject);
QJsonDocument doc(recordsArray);

QString filename="../test.json";
 QFile file( filename );
if ( file.open(QIODevice::ReadWrite) )
{
    QTextStream stream( &file );
    stream << doc.toJson() << endl;
}
```

در ادامه برای ساخت ارایه در جی‌سان از کلاس `QJsonArray` استفاده شده است و برای ارسال مقادیر به لیست مربوطه کافی است از متدهای `push_back` استفاده شود که مقادیر را دریافت و توسط `insert` مربوط به کلاس `QJsonObject` با شناسه مربوطه ذخیره سازی می‌شود. در نهایت آرایه مربوطه و سند آن ایجاد و توسط کلاس `QFile` در محل مربوطه ذخیره سازی خواهد شد.

## معرفی کلاس QSetting

این کلاس تنظیمات مستقل از پلتفرم را فراهم می‌کند. برای مثال برنامه‌ها معمولاً نیاز دارند که تنظیمات مربوط به اندازه، موقعیت و ویژگی‌های خودشان را که توسط کاربر اعمال می‌شود را ذخیره سازی نمایند تا هر بار اجرا می‌شود نیازی به تنظیم مجدد آن نباشد. در کیوت کلاسی برای سهولت این کار ایجاد شده است با نام `QSetting` که امکان آن را فراهم می‌سازد. به طور پیشفرض زمانی که از این کلاس استفاده می‌شود باید نام شرکت سازنده محصول، و نام محصول را به درستی در این کلاس تنظیم کنیم. به عنوان مثال کد زیر نمونه‌ای از آن را نمایش می‌دهد:

```
QSettings settings("Genyleap", "Panamera");
```

کد فوق نشان میدهد که تنظیمات مربوطه برای شرکت جنیلیپ برای محصول Panamera صورت گرفته است. تمامی اشیاء موجود در `QSetting` میتوانند درپیشه و یا استک ایجاد شوند. ساخت و از بین بردن اشیاء در کلاس `QSetting` در بخش‌های مختلف نرمافزار خود استفاده کنید پیشنهاد می‌شود از روش `QSetting` داشته باشید که اگر شما می‌خواهید از قابلیت‌های `QCoreApplication::setOrganizationName` و `QCoreApplication::setApplicationName` استفاده کنید و سپس عمل ساخت را انجام دهید.

```
QCoreApplication::setOrganizationName("Genyleap");
QCoreApplication::setOrganizationDomain("genyleap.com");
QCoreApplication::setApplicationName("Panamera");
QSettings mySetting;
```

کلاس `QSetting` دارای صفاتی مانند `beginGroup` و `endGroup` است که نشان می‌دهد بازه آغاز و پایان یک دستور تنظیمات چطور آغاز و به پایان می‌رسد. کد زیر مثالی از آن را تعریف می‌کند:

```
QSettings mySetting;

mySetting.beginGroup("mainwindow");
mySetting.setValue("size", win->size());
mySetting.setValue("fullscreen", win->isFullScreen());
mySetting.endGroup();
```

ممکن است این کار برنامه نویسان حرفه‌ای کلاسی برای ساخت و تنظیم فایل ini ایجاد می‌کند که در این میان کیوت این امکان را برای همگان فراهم ساخته است. این بستگی به آن دارد که شما تا چه مقدار بخواهید تحت ساختار کیوت برنامه خود را سفارشی سازی کنید. بنابراین تحت کیوت کد زیر می‌تواند تنظیماتی را برای ما فراخوانی کند:

```
QSettings settings("../test.ini", QSettings::IniFormat);
settings.beginGroup("DATA_ONE");
const QStringList childKeys = settings.childKeys();
foreach (const QString &childKey, childKeys)
    qDebug() << settings.value(childKey);
settings.endGroup();
```

در این بخش پاراکتر اول نوع setting از کلاس QSetting مسیر فایل مربوط به تنظیمات است و پارامتر دوم نشانگر نوع قالب ذخیره سازی است که می‌تواند شامل NativeFormat برای قالب پیش فرض بومی و IniFormat از قالب مشهور ini پشتیبانی می‌کند. در ادامه با مشخص سازی ابتدای گروه تنظیمات با beginGroup و سپس معرفی کلیدهای فرزند تحت نوع QStringList مشخص سازی شده است که در ادامه آن رشته کلیدها توسط متدهای value دریافت می‌شوند. در آخر نیز توسط endGroup به پایان گروه تنظیمات اشاره شده اشاره می‌شود.

در ادامه برای دریافت تنها مقدار هریک کافی است از متدهای زیر استفاده کنیم:

```
qDebug() << settings.value(childKey).toString();
```

خروجی مربوط به آن به صورت زیر خواهد بود:

```
"value1"  
"value2"
```

## سفارشی سازی فایل pro. در پروژه تحت کیوت

همانطور که می‌دانید فایل pro. مهمترین فایل هریک از پروژه‌های تحت کیوت برای سی‌پلاس‌پلاس است و بیشتر جهت پیکربندی‌های اولیه در رابطه با سورس، مازول‌ها، پرچم (فلگ)‌ها و موارد مشابه است. در نظر داشته باشید به دلیل خاصیت چند سکویی بودن این کتابخانه که از ذات طبیعی C++ پشتیبانی می‌کند تنظیمات مربوط به هر یک از پلتفرم‌ها به صورت جداگانه در این فایل صورت می‌گیرد. بنابراین فرض کنید برنامه‌ای نوشته‌اید که قرار است بر روی پلتفرم‌های مختلفی مانند iOS، Android، Windows و Linux اجرا شود که برای هر یک نیاز خواهد بود جداگانه تنظیماتی را در فایل pro. ایجاد نماییم. برای این کار به روش‌هایی اشاره می‌شود که در توسعه برنامه‌های حرفه‌ای کمک بسزایی خواهد داشت.

```
unix {  
    // تعریف تنظیمات مربوط به یونیکس / لینوکس  
}  
qnx {  
    // تعریف تنظیمات مربوط به یونیکس / بلک بری  
}  
android {  
    // تعریف تنظیمات مربوط به اندروید  
}  
wi32 {  
    // تعریف تنظیمات مربوط به ویندوز  
}  
macx {  
    // تعریف تنظیمات مربوط به مکینتاش
```

توسط کلمات کلیدی مربوط به هر یک از پلتفرم‌ها می‌توان تنظیمات اختصاصی هر یک را در یک پروژه سفارشی سازی کرد. برای مثال کد زیر نشان می‌دهد که چطور ممکن است نسخه زبان برنامه‌نویسی را برای هریک از سکوها متفاوت تعریف کرد.

```
unix {  
    CONFIG += c++11  
}  
qnx {  
    CONFIG += c++11  
}  
android {  
    CONFIG += c++14  
}  
wi32 {  
    CONFIG += c++17  
}  
macx {  
    CONFIG += c++17  
}
```

این امکان در توسعه انواع کتابخانه‌ها برای سکوهای مختلف بسیار مفید خواهد بود.

تنظیم پروژه از نوع کتابخانه (Lib)، جهت پیکربندی پروژه در این قالب کافی است در فایل pro. مقدار TEMPLATE را برابر lib قرار دهید تا بعد از کامپایل در فرمت خروجی کتابخانه‌ای با پسوندهای .dll. در ویندوز، .dylib. در مک و .so. در یونیکس (لینوکس) تولید شود.

```
TEMPLATE = lib
```

در صورتی که نیاز به این قالب نباشد و هدف تولید یک فایل اجرایی باشد کافی است مقدار TEMPLATE را برابر app قرار دهید.

## مقایسه انواع حالت‌های کامپایل Release و Debug

در تمامی محیط‌های توسعه انواع حالت‌های Debug و Release مشاهده می‌شود. در این میان کیوت از این قاعده پشتیبانی می‌کند. نکته‌ای که قابل توجه است آن است که در زمان کامپایل بر روی حالت Debug نتیجه نهایی فایل‌ها با حجم بیشتری مواجه شده و برای اجرا و آزمایش آن نیاز به فایل‌های پیش‌نیاز با خاتمه دهنده کلمه کلیدی `l` در ایستگاه‌های یونیکس و debug در ایستگاه‌های ویندوز مشخص می‌شوند.

برای مثال در مک و یا لینوکس اگر برنامه ما نیازمند راهانداز MySQL باشد باید فایل `libqsqlmysql_debug.dylib` قابل فراخوانی باشد تا برنامه بتواند از آن استفاده کند. و یا در ویندوز `QSqlmysql.dll` مورد شناسایی خواهد بود. در غیر این صورت اگر برنامه در حالت Release منتشر شود کافی است فایل‌های پیش‌نیازی که از این قاعده پیروی نمی‌کنند را در کنار برنامه خود فراهم سازید.

نکته: اندازه فایل‌های حالت Debug شده بسیار بیشتر از حالت Release نهایی است! بنابراین توجه داشته باشید که برنامه نهایی شما دارای حجم بسیار پایینتری نسبت به نسخه آزمایشی تحت Debug خواهد بود.

### در رابطه با حالت داینامیک (Dynamic) توضیح مختصر:

در این حالت شما در بسیاری از موارد برای توسعه نرم‌افزار در دو حالت OpenSource و انحصاری قادر خواهید بود.

مزایا:

- معمولاً برنامه برای کاربر نهایی یا همان End-User یک بسته جمع و جور و کامل را فراهم می‌کنید و همچنین فایل اجرایی در کم حجم‌ترین و فشرده‌ترین حالت خارج خواهد شد.
- کتابخانه‌های Qt را شما می‌توانید بدون تغییر و کامپایل مجدد پروژه آن‌ها را برای توسعه دهنده‌گان بازخورد داده و یا به روز رسانی نمایید و حتی آن‌ها را تغییر دهید.
- نیاز به منابع سخت افزاری بسیار کمی است برای مثال اشغال حافظه Ram بسیار کمتر از حالت Static است.

معایب:

برای مطمئن شدن از کارائی درست برنامه در هر سیستم و هر بستری که نیاز است بارها و بارها از هر جوانبی برنامه را نسبت به کتابخانه بررسی نمایید تا زمانی که مورد استفاده توسط کاربر یا همان End-User قرار می‌گیرد بدون بروز مشکل یا خطای اجرا شود. معمولاً بر روی سیستم‌های Linux نیز باید کتابخانه‌های به درستی نصب شوند.

شما باید مطمئن شوید که تمام کتابخانه‌های مورد نیاز در سیستم هدف (End User) در دسترس هستند در صورتی که بر روی سیستم مورد نظر در دسترس نیستند باید راه حل مناسبی برای ارائه کتابخانه‌های مورد نظر فراهم نمایید و خدمات آن را در اختیار کاربر قرار دهید.

### در رابطه با حالت استاتیک (Static) توضیح مختصر:

در این حالت شما می‌توانید مطمئن شوید که برنامه شما در هر سیستمی بدون نیاز به پیش نیازی قابل اجرا خواهد بود.

**مزایا:**

- معمولاً برنامه برای کاربر نهایی یا همان End user یک بسته جمع و جور و کامل را فراهم می‌کنید.
- برنامه شما می‌توانید مستقل از هر نسخه از کتابخانه‌های موجود بر روی سیستم کاربر برنامه را اجرا کنید حال چه در کیوت ۴.۶ باشد و چه در کیوت ۵.۱۳ هیچ تداخلی نخواهد داشت.

**معایب:**

- درخواست‌های برنامه شما به کتابخانه بسیار زیاد و سنگین خواهد بود زیرا کتابخانه‌ها نیز به برنامه شما متصل و لینک شده هستند.
- ممکن است برای رفع مشکلات کتابخانه و تغییر، به روز رسانی و ... مجبور به کامپایل مجدد برنامه شوید.
- مصرف منبع Ram در صورت درخواست‌های پی در پی و چند گانه بسیار زیاد خواهد بود.
- در حالت Runtime شما قادر به اجرای plugins توسط QPluginLoader نخواهید بود.

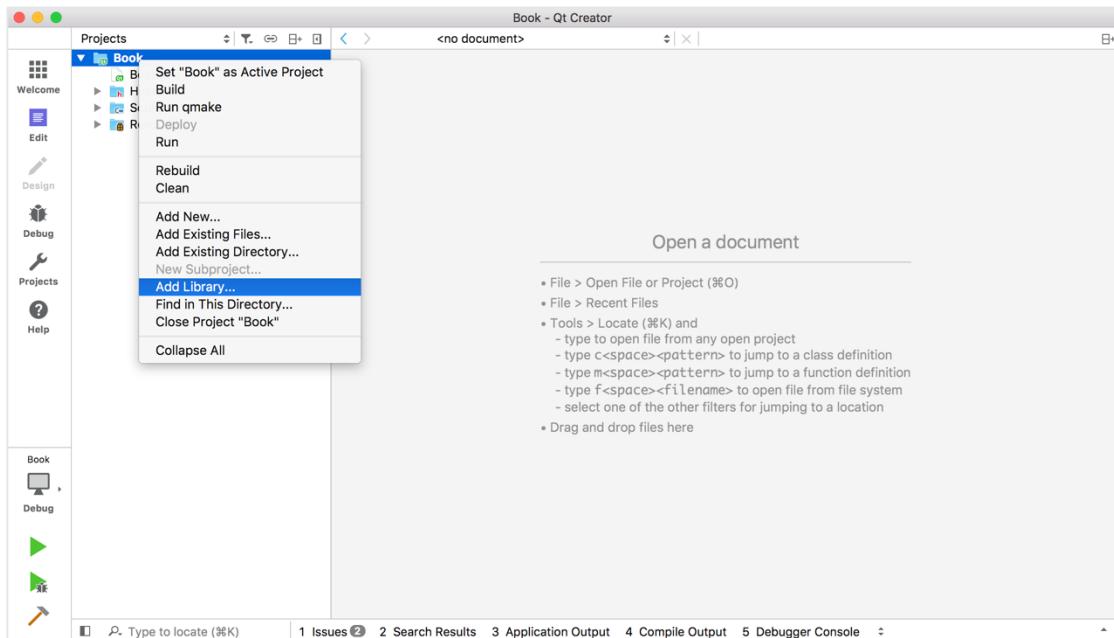
برای کامپایل به صورت استاتیک (ایستا) بدون داشتن لیسانس مربوطه از طرف شرکت کیوت مجاز نمی‌باشد.

## نحوه افزودن کتابخانه‌های دیگر C++ در محیط توسعه Qt Creator

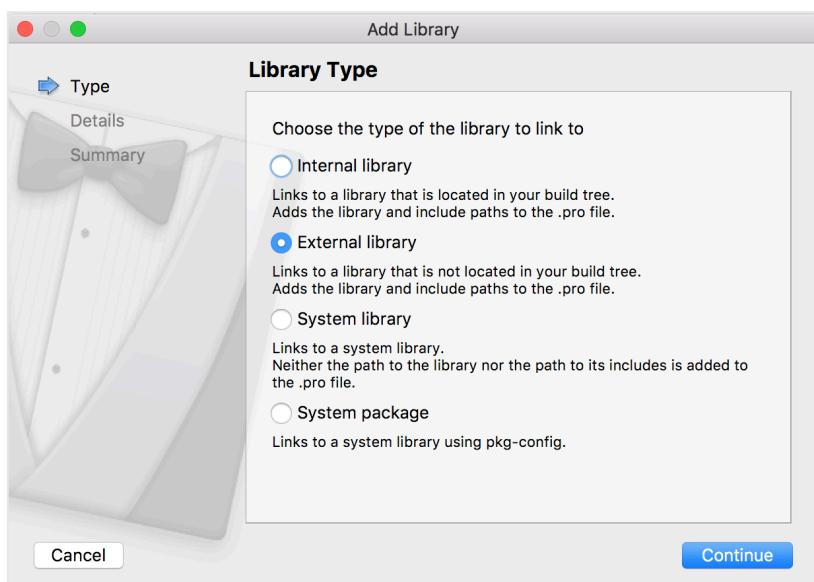
معمولًاً برنامه‌نویسان C++ مشتاق هستند از کتابخانه‌های سفارشی خودشان و یا کتابخانه‌هایی که مورد علاقه آن‌ها است استفاده کنند. که این امر در محیط Qt Creator نیز همانند دیگر محیط‌های توسعه در C++ امکان‌پذیر است، برای مثال می‌توانیم کتابخانه‌ای مانند Boost, Poco و غیره را همراه با Qt مورد استفاده قرار دهیم. زبان C/C++ یکی از قابلیت‌هایی که نسبت به زبان‌های دیگری دارد نامحدود بودن استفاده از کتابخانه‌های iostream این زبان است که به صورت پیشفرض کتابخانه‌های استاندارد و از قبل تعریف شده در زبان C++ قابل استفاده هستند مانند کلاس‌های std::string و ... که کاملاً پیشفرض همراه با هسته این زبان ارائه شده است.

ابتدا کتابخانه مورد نظر را دریافت کرده و در مسیر مقابله به صورت دلخواه قرار میدهیم : C:/Library/poco-{version} (در این مثال من از Poco نسخه ۱.۷.۸ استفاده شده است. قبل از هر چیزی باید در نظر داشته باشید که معمولاً هر کتابخانه‌ای دارای پوشش‌های lib و include است هر نوع کتابخانه‌ای که دانلود می‌کنید (به جز انواع خاصی که فقط وابسته به سرآیندها هستند) این دو گزینه را باید داشته باشد که بعد از کامپایل کتابخانه فایل lib, dylib و dll به ترتیب در لینوکس، مک و ویندوز ایجاد خواهد شد.

در این مثال به دلیل پیچیده بودن کتابخانه POCO (پوکو) را انتخاب کردہ‌ایم چون شامل کتابخانه‌های مربوط به Net و ... است که برای آموزش هدف کار با Net است بنابراین پوشش‌های include و libs در کنار هم نخواهند بود، لذا include مربوط به هر کتابخانه در داخل خودش قرار گرفته است. بر روی پروژه طبق تصویر راست کلیک کنید و گزینه Add library را بزنید.

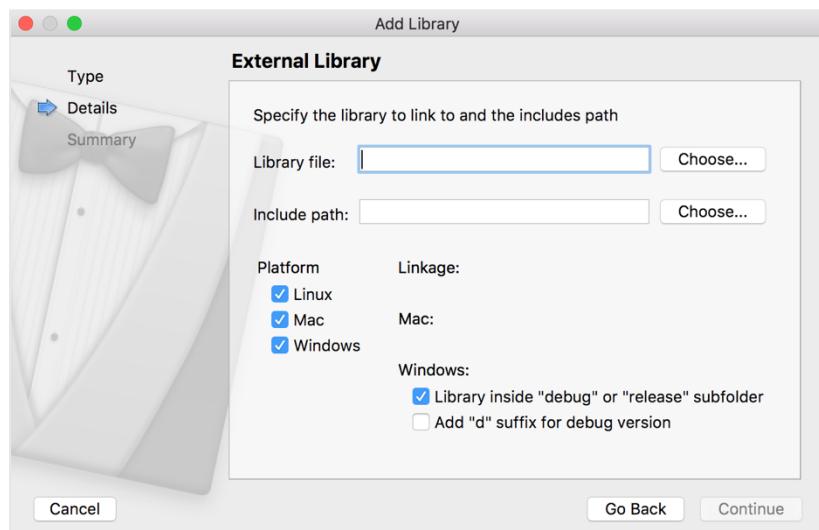


سپس در بخش بعد از آن بخش افزودن کتابخانه را مشاهده خواهید کرد:

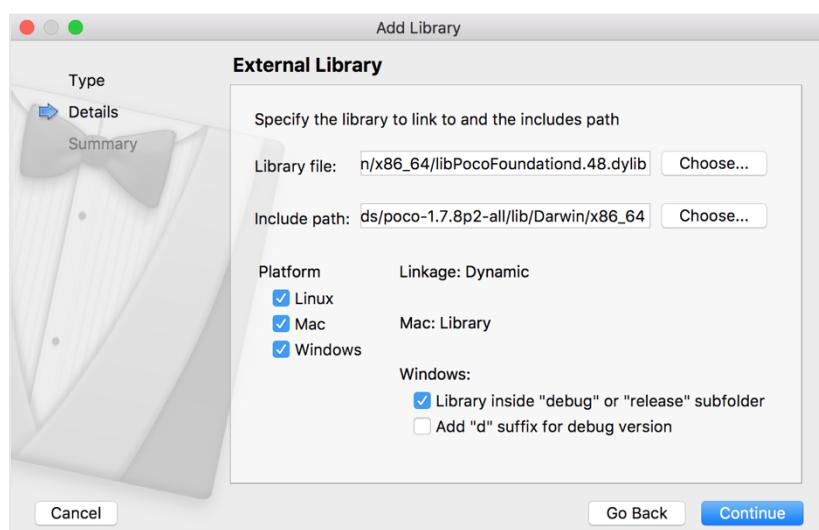


۱. گزینه Internal Library مربوط است برای زمانی که شما کتابخانه را در داخل پروژه خودتان ایجاد کرده‌اید که معمولاً مسیر واقع توسط فایل `.pro` مشخص خواهد شد به طور کلی کتابخانه‌های داخلی و غیر External را می‌توانید از این قسمت شناسایی کنید.
۲. گزینه External library، زمانی استفاده می‌شود که کتابخانه‌ما در مسیر پروژه نباشد یا به طور کلی در داخل پروژه قرار نگرفته باشد دقیقاً برعکس گزینه اول که همان کتابخانه خارجی یا خارج از پروژه است و بخشی از سیستم عامل هستند.
۳. گزینه System library هم مربوط می‌شود به کتابخانه‌های سیستمی.
۴. گزینه‌ای خواهیم داشت در محیط‌های Unix که به نام Package Library قابل مشاهده خواهد بود، این گزینه زمانی مورد استفاده قرار می‌گیرد که شما نیاز دارید کتابخانه را از طریق سرویس `pkg-config` در ایستگاه‌های Unix Linux و macOS می‌باشند تنظیم کنید.

در حالت عادی از External Library استفاده خواهیم کرد، بنابراین گزینه External Library را انتخاب و Next را اعمال کنید:



در این قسمت گزینه‌هایی که مربوط به پیکربندی و مشخص کردن مسیر کتابخانه‌ای است را مشاهده می‌کنید، که از قبل کامپایل و خروجی‌های lib و یا dll آنها مشخص شده است.



بنابراین گزینه‌ها وظایف زیر را دارند:

گزینه Library file مربوط است به مسیر فایل‌های مربوط به libs موجود در کتابخانه، برای مثال: مسیر lib برای کتابخانه Poco در ایستگاه یونیکس (مک) مسیر /poco-{version}... بوده و در پلتفرم ویندوز: C:\..\poco-{version}\ است، توجه داشته باشید این مسیر بر اساس ریشه فایل شما بر روی سیستم عامل که بعد از دانلود و کامپایل کتابخانه اعمال کرده‌اید مشخص می‌شود. بنابراین، بر روی گزینه Browse کلیک کرده و این مسیر را با انتخاب فایل مورد نظر مثلا PocoNetd.lib مشخص کنید که مسیر نهایی آن چیزی به صورت زیر خواهد بود:

Windows : C:\...\poco-{version}\lib\PocoNetd.lib

Unix / Linux: .../... /poco-{version}/lib/libPocoFoundationd.48.dylib

گزینه Include path مربوط است به پوشش کتابخانه که شامل فایل‌های h. یا همان سرآیند بنابراین برای اینکه مربوط به کلاس‌های Net را انتخاب کنیم به مسیر C:\Library\poco-{version}\Net\include : مراجعه می‌کنیم که شامل Include های مشخص برای فایل Net است.

- گزینه Platforms مشخص کننده مسیر در فایل .pro برای نوع سیستم‌عامل است.
- گزینه Dynamic - Linkage یا Static Framework و گزینه Dynamic مشخص کننده کتابخانه از نوع است.
- و در آخر گزینه‌ای که مختص ویندوز است مشخص می‌شود.

تنظیمات را به پایان رسانده و بعد از این مرحله کدهای تولید شده در فایل .pro. به صورت زیر خواهد بود :

```
win32:CONFIG(release, debug|release): LIBS += -L$$PWD/../../../../../Downloads/poco-1.7.8p2-all/lib/Darwin/x86_64/release/ -lPocoNetd.48
else:win32:CONFIG(debug, debug|release): LIBS += -L$$PWD/../../../../../Downloads/poco-1.7.8p2-all/lib/Darwin/x86_64/debug/ -lPocoNetd.48
else:unix: LIBS += -L$$PWD/../../../../../Downloads/poco-1.7.8p2-all/lib/Darwin/x86_64/ -lPocoNetd.48

INCLUDEPATH += $$PWD/../../../../../Downloads/poco-1.7.8p2-all/Net/include
DEPENDPATH += $$PWD/../../../../../Downloads/poco-1.7.8p2-all/Net/include
```

در ادامه فایل .pro را ذخیره کرده و روی گزینه build و بعد Run qmake کلیک را اعمال کنید تا پروژه با تغییراتی که داده شده است پیکربندی و آماده شود. برای آزمایش کد زیر را مورد آزمایش قرار دهید:

```
#include <QCoreApplication>
#include <Poco/Net/Net.h>
#include <Poco/Net/MailMessage.h>

using namespace Poco::Net;

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    MailMessage mail;

    return a.exec();
}
```

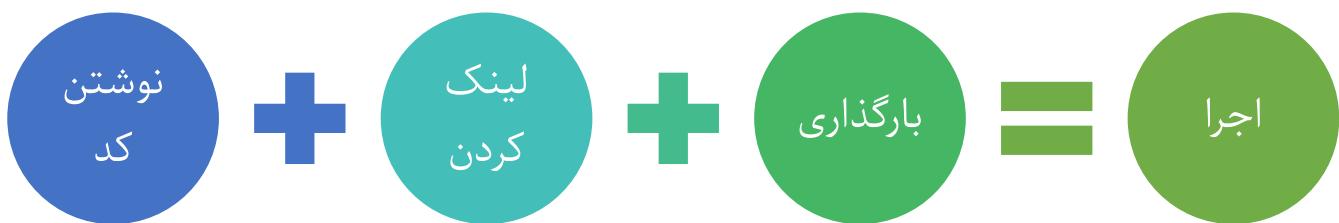
## فرق بین کامپایل استاتیک و داینامیک

قبل از اینکه تفاوت بین ایستا (استاتیک) - Static و پویا (داینامیک) - Dynamic را بدانید لازم است در رابطه با چرخه زندگی نوشتمن یک برنامه و اجرای آن آشنا شویم. هر برنامه برای اولین بار توسط یک محیط توسعه (Editor) یا IDE توسط برنامه‌نویسان انتخاب و به صورت فایل متنی قابل ویرایش است. سپس فایل متنی که شامل کدهای نوشته شده توسط برنامه‌نویس تحت زبان برنامه‌نویسی مانند C, C++ و غیره... می‌باشد توسط کامپایلر به کد شیء ای تبدیل می‌شود که ماشین بتواند آن را درک کرده و اجرا کند.

برنامه‌ای که ما می‌نویسیم ممکن است به عنوان یک مورد توسط دیگر برنامه‌ها یا کتابخانه‌هایی از برنامه‌ها مورد استفاده قرار بگیرد برقراری ارتباط (پیوند کردن) یا همان لینک کردن پروسه‌ای است که برای اجرای موفقیت آمیز برنامه‌های نوشته شده ما بکار می‌رود برقراری ارتباط بین ایستا و پویا دو پروسه‌ای از جمع آوری و ترکیب فایل‌های شیئی‌های مختلفی است که به منظور ایجاد یک فایل اجرایی می‌باشند. در این بخش ما تصمیم بر این داریم تا تفاوت بین آن‌ها را با جزئیات مورد بررسی قرار دهیم.

عمل پیوند یا ترکیب در زمان کامپایل انجام شود، در واقع زمانی که کد منبع به زبان ماشین ترجمه می‌شود، در زمان بارگذاری، زمانی که برنامه در داخله حافظه بارگذاری می‌شود، و حتی زمان اجرای آن توسط برنامه صورت می‌گیرد این عمل زمان پیوند و یا ترکیب (اتصال) است. در نهایت این فرآیند توسط برنامه‌ای اجرا می‌شود که به آن لینکر - پیوند دهنده (ترکیب کننده) می‌گویند. اتصال دهنده‌ها به عنوان ویرایشگر لینک نیز معرفی می‌شوند. لینک شدن (پیوند شدن) به آخرین مرحله از کامپایل می‌گویند.

در زبان علمی اصطلاح لینکر یا Linker معروف است اما در زبان فارسی بهترین گزینه مربوطه را می‌توان با عنوان اتصال دهنده، پیوند دهنده ترکیب کننده نام برد. همه آن‌ها نشانگر یک هدف به منظور ترکیب اشیاء با یکدیگر هستند که در مرحله کامپایل صورت می‌گیرد. پس از ایجاد پیوند در برنامه، برای اجرای آن برنامه باید داخل حافظه منتقل شود. در انجام این کار باید آدرس‌هایی برای اجرای داده‌ها و دستور العمل‌ها اختصاص یابد. به طور خلاصه روند زیر می‌تواند به عنوان چرخه زندگی یک برنامه خلاصه شود (نوشتن - لینک کردن - بارگذاری - اجرا)



در ادامه تفاوت‌های عمدۀ ارتباط بین استاتیک و داینامیک در یک پروژه آمده است.

#### استاتیک (ایستا)

- ارتباط به روش استاتیک فرآیندی است که تمامی مازول‌ها و کتابخانه‌های برنامه در فایل اجرایی نهایی کپی می‌شوند. این روش توسط لینکر (لينک کننده) در مرحله آخر کامپایل انجام می‌شود. اتصال دهنده - لینکر طبق روال ترکیبی کتابخانه‌ها را با کد برنامه و همراه مراجع - منابع خارجی ترکیب کرده و برای تولید یک بارگذاری مناسب در حافظه آماده سازی می‌کند. زمانی که برنامه بارگذاری می‌شود، سیستم‌عامل محلی را در حافظه به صورت یک فایل اجرایی که شامل کدهای اجرایی و داده‌ها است مشخص می‌کند.

- ارتباط به شیوه استاتیک توسط برنامه‌ای با نام لینکر انجام می‌شود که در آخرین مرحله فرآیند کامپایل یک برنامه صورت می‌گیرد. لینکرها نیز به عنوان ویرایشگر پیوند نیز عنوان می‌شوند.

- فایل‌های استاتیک به طور قابل توجهی دارای اندازه بسیار بزرگی هستند زیرا برنامه‌های خارجی و کتابخانه‌های لینک شده همه در یک جا و در فایل نهایی اجرایی جمع آوری شده‌اند.

- در اتصال استاتیک اگر هر یک از برنامه‌های خارجی تغییر کرده باشد باید آن‌ها دوباره کامپایل شوند و مجدداً عمل اتصال صورت گیرد در غیر اینصورت هیچ تغییری در به روز رسانی‌های مرتبط با فایل اجرایی مشاهده نخواهد شد.

- برنامه‌های استاتیکی زمان بارگذاری ثابتی در هر بار اجرای برنامه در حافظه را در نظر می‌گیرند. و زمانی که برای بارگذاری طول می‌کشد ثابت است.
- برنامه‌هایی که از کتابخانه‌های استاتیکی استفاده می‌کنند معمولاً سریعتر از برنامه‌هایی هستند که کتابخانه‌های آن‌ها به صورت پویا می‌باشد.

#### حالت دینامیک (پویا)

- در ارتباط پویا نام کتابخانه‌های خارجی (کتابخانه‌های به اشتراک گذاری شده) در فایل اجرایی نهایی قرار داده شده‌اند نه خود کتابخانه. در حالی که ارتباط واقعی در زمان اجرا در هر دو فایل در حافظه قرار می‌گیرند. اتصال پویا این اجازه را می‌دهند تا برنامه‌های متعددی به صورت یک مازول کپی شده و قابل اجرا مورد استفاده قرار بگیرد.
- اتصال پویا برخلاف اتصال استاتیک در زمان اجرا توسط سیستم‌عامل انجام می‌شود.
- در اتصال پویا فقط یک نسخه از کتابخانه به اشتراک گذاری شده در حافظه نگه داری می‌شود. این به طور قابل توجهی اندازه برنامه‌های اجرایی را کاهش می‌دهد، در نتیجه صرف‌جویی در حافظه و فضای دیسک صورت خواهد گرفت.
- در اتصال پویا برخلاف اتصال استاتیک نیازی به کامپایل کامل پروژه نمی‌باشد در صورتی که لازم باشد تغییراتی در هر یک از فایل‌ها صورت بگیرد تنها کافی است آن را کامپایل و در کنار برنامه قرار دهید. این یکی از بزرگترین مزیت‌های کامپایل دینامیکی است.
- در اتصال پویا زمان بارگذاری برنامه در حافظه ممکن است کاهش یابد. این در صورتی است که کتابخانه‌های مشترک در حافظه بارگذاری شده‌اند.
- برنامه‌هایی که از کتابخانه‌های مشترک استفاده می‌کنند معمولاً کنترل از برنامه‌هایی هستند که از کتابخانه‌های استاتیکی استفاده می‌کنند.
- برنامه‌های پویا وابسته به داشتن کتابخانه‌های سازگار هستند. اگر کتابخانه تغییر یابد (برای مثال، یک کامپایلر جدید منتشر شود ممکن است کتابخانه را تغییر دهد)، در این صورت ممکن است برنامه مجددأ تحت کتابخانه جدید باز نویسی و به روز رسانی شوند. اگر کتابخانه از روی سیستم حذف شود، برنامه‌ای که وابسته آن کتابخانه می‌باشد دیگر کار نخواهد کرد.

### نحوه تهیه خروجی و گسترش (Deployment) در Qt

با توجه به اینکه روش گسترش یا اجرای برنامه‌ها در هر بستری متفاوت است، برای این منظور توضیحاتی در رابطه با نحوه اجرا بر روی هر پلتفرم را به صورت جداگانه توضیح می‌دهیم.

#### توسعه و اجرای برنامه بر روی بستر ویندوز:

در این محیط نسبت به نوع و نسخه Qt و کامپایلری که مورد استفاده قرار گرفته است باید توجه داشته باشیم که هنگام کامپایلر و خروجی گرفتن متناسب با سیستم مقصد آن را تهیه کنیم، برای مثال نوع معماری یعنی x64 یا x86 بودن یک سیستم بسیار مهم است.

**نکته:** اجرا شدن برنامه‌های ۶۴ بیتی بر روی سیستم‌های ۳۲ بیتی امکان‌پذیر نیست.

در ابتدا بعد از ایجاد پروژه مهم است که مشخص شود نوع و نسخه کامپایلر و حتی حالت یا همان (Mode) کامپایلر بر چه اساسی تنظیم شده است، با توجه به این موارد فایل‌های مورد نیاز و نحوه اجرای برنامه متفاوت است.

## مواردی که باید به آن‌ها هنگام کامپایل توجه کنیم:

۱. مشخص سازی نوع کامپایل برنامه حالت یا همان Mode ای که برنامه روی آن ساخته می‌شود، اگر برنامه بر روی Debug ساخته می‌شود تمامی موارد بعدی بر اساس دیباگ تعیین و در غیر اینصورت بر اساس نوع Release مشخص خواهد شد.
۲. نوع معماری خروجی در برنامه، باید توجه داشته باشد برنامه‌های ۳۲ بیتی توسعه کامپایلرهای x86 یا ۶۴ بیتی و برنامه‌های ۶۴ توسعه کامپایلرهای x64 که خود نیازمند سیستم و بستر برنامه‌نویسی می‌باشند که ۳۲ بیتی هستند، یعنی اگر نیاز باشد برنامه شما بیتی کامپایل شود ابتدا باید سیستم‌عامل و نسخه کامپایلر محیط توسعه از آن پشتیبانی کند.
۳. انواع مازول‌های استفاده شده در کتابخانه Qt مهم است، به عنوان مثال در حالت عادی مازول Qt5Core نیاز است ولی اگر در پروژه شما از مازول‌های دیگری مانند Network استفاده شده باشد در این حالت نیاز خواهید داشت فایل یا مازول مربوط به آن را وارد برنامه کنید که شامل Qt5Network است، لیست کاملی از مازول‌ها را بر اساس نیاز در ادامه مشخص خواهیم کرد که بر چه اساسی چه نوع مازول و چه فایلی باید همراه برنامه موجود باشد.

## شروع کامپایل و گسترش برنامه:

معمولًاً نسخه‌های آزمایشی یک محصول در حالت Debug جهت بررسی و آنالیز خطاهای موجود در آن است که توسعه‌دهنده یا افرادی که می‌توانند در باگ گیری آن همیاری نمایند استفاده خواهد کرد، بنابراین بر فرض اینکه ما قرار است یک نسخه استاندارد و نهایی از محصول را در اختیار کاربر قرار دهیم از حالت Release استفاده خواهیم کرد.

در بخش Projects می‌توان نوع کامپایلر و مسیر خروجی از آن را مشخص کرد، وقت کنید که در این بخش قسمت Build بر روی حالت Release باشد، در این مثال ما از کامپایلر MSVC2017 و نسخه ۶۴ بیتی آن استفاده کرده‌ایم که مسیر خروجی آن مشخص است.

همانند مک و لینوکس در ویندوز نیز ابزاری با نام QTDIR/bin/windployqt وجود دارد که در مسیر QTDIR/bin/windployqt است. توسعه این ابزار می‌توان برنامه را در قالب یک پکیج جمع آوری و مستقر ساخت.

همانند مک و لینوکس در ویندوز نیز ابزاری با نام QTDIR/bin/windployqt وجود دارد که در مسیر QTDIR/bin/windployqt است. توسعه این ابزار می‌توان برنامه را در قالب یک پکیج جمع آوری و مستقر ساخت.

برای مثال ما برنامه‌ای ساخته‌ایم که در مسیر مورد نظر /C:/Users/Compez/Desktop/MyAppRoot است. با دستور cd به مسیر فوق خواهیم رفت:

```
cd C:/Qt/Qt5.9.0/5.9/msvc2017_64/MyAppRoot
```

البته قرار است در این مسیر خروجی فایل بعد از کامپایل ایجاد شود که با غیر فعال‌سازی امکان Shadow Build این ممکن خواهد شد که فایل مربوطه در مسیر ریشه برنامه ایجاد شود. با فرض اینکه بعد از کامپایل فایل MyAppRoot.app در مسیر ذکر شده موجود باشد دستور زیر را در ترمینال وارد خواهیم کرد:

```
C:/Qt/Qt5.9.0/5.9/msvc2017_64/bin/windployqt MyAppRoot.exe
```

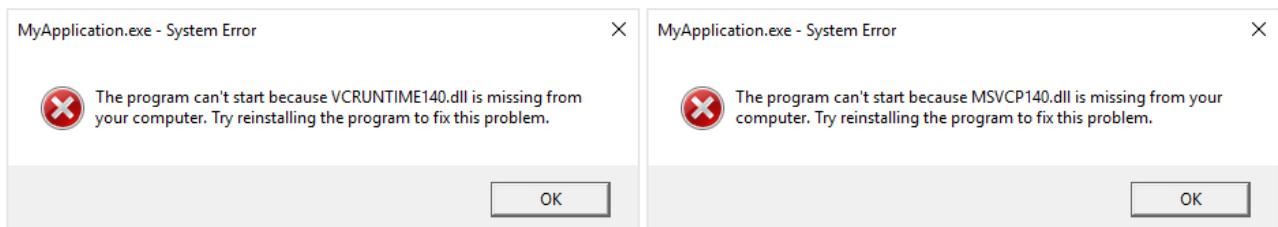
دقیق کنید که اگر نیاز باشد با استفاده از گزینه‌های موجود در ابزار برنامه خود را مستقر سازید کافی است دستور ایجاد را به صورت زیر وارد کنید:

```
C:/Qt/Qt5.9.0/5.9/msvc2017_64/bin/windeployqt MyApplication.app -verbose=3 -no-plugins
```

در ویندوز برخلاف ایستگاه‌های یونیکس فراهم آوردن تمامی فایل‌ها در کنار برنامه صورت خواهد گرفت. اما بعد از اجرای دستور فوق برنامه به تنها‌یابی قابل اجرا نخواهد، لذا فایل‌های msvcp140.dll و vcruntime140.dll نیاز هستند تا در کنار برنامه قرار گیرند. این فایل‌ها در تمامی نرم‌افزارهای بزرگ در کنار برنامه موجود هستند مگر اینکه به صورت جدا پکیج مربوط به آن را نصب کنید که توصیه نمی‌شود. توجه داشته باشید که فایل‌هایی که قبل از پسوند .dll. آخر حرف آن‌ها به **d** ختم می‌شود نشانگر آن است که مربوط به نسخه دیباگ هستند. در صورتی که در حالت Release برنامه خود را کامپایل می‌کنید فایل‌هایی را در کنار برنامه خود قرار دهید که حرف آخر آن‌ها به **d** ختم نشده باشد. برای مثال فایل **QtCore.dll** مخصوص نسخه دیباگ بوده و فایل **QtCore.d.dll** مخصوص نسخه ریلیز.

نکته: در نظر داشته باشید که همین روند برای کیوت نسخه MinGW نیز صدق می‌کند.

بعد از کامپایل برنامه و اجرای خروجی آن در ویندوزی که بر روی آن Qt و C++ نصب نیست مسلماً با خطاهای زیر مواجه خواهیم شد:



خطاهای فوق بیانگر این است که فایل‌های فوق در کنار پروژه یا در هسته سیستم‌عامل پوشش windows/SysWow64 و یا windows/system32 داشته باشند. نصب نشده است که در ادامه برای حل این خطاهای راهکار ارائه داده شده است.

بنابراین به مسیر زیر بروید :

C:/Program Files (x86)/Microsoft Visual Studio 2017/Enterprise/VC/Redist/14.x.x/onecore/x64/Microsoft.VC150.CRT  
سپس فایل‌های موجود در پوشش را کپی و در کنار برنامه قرار دهید در این صورت برنامه بدون هیچ خطایی اجرا خواهد شد. مگر اینکه به جز کتابخانه **Qt** و **STL** از کتابخانه‌های دیگر استفاده کرده باشید که در این صورت هم باید فایل‌های مربوط به آن‌ها را در کنار برنامه قرار دهید.

بسته‌های نصبی برای برنامه‌های نوشته شده توسط C++ که در زیر پیشنهاد شده‌اند:

فراهم سازی آرشیو کامل پیش نیازات برای برنامه‌های C++ با توجه به کامپایلر صورت می‌گیرد، با توجه به اینکه در این مثال ما از کامپایلر MSVC2017 نسخه ۶۴ بیتی استفاده کرده‌ایم باید فایل‌های مربوط به این کامپایلر در اختیار فایل اجرایی قرار بگیرد که این به دو روش امکان پذیر است، اگر توجه کرده باشید معمولاً بعد از نصب سیستم‌عامل شروع نرم‌افزارهایی که تحت C++ نوشته شده‌اند معمولاً در مراحل نصب پکیج مورد نیاز را بر اساس نوع و نسخه کامپایلر نصب می‌کنند که شامل موارد زیر است:

توضیحات	نام پکیج همراه با نسخه آن
بسته مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2005	Microsoft C++ Redistributable 2005
	Microsoft C++ Redistributable 2008

بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2008	
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2010	Microsoft C++ Redistributable 2010
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2012	Microsoft C++ Redistributable 2012
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2013	Microsoft C++ Redistributable 2013
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2015	Microsoft C++ Redistributable 2015
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2017	Microsoft C++ Redistributable 2017
بستهٔ مربوط به کتاب به پیش‌نیازات کتابخانه‌های استاندارد C++ در کامپایلر MSVC2019	Microsoft C++ Redistributable 2019

با توجه به جدول بالا باید در نظر داشته باشید تمامی کتابخانه‌های استاندارد C و C++ بر اساس نسخه کامپایلر در پکیج‌های ذکر شده موجود هستند، که معمولاً با نصب آن‌ها فایل‌های بسیاری در رابطه با کتابخانه‌ها بر روی سیستم‌عامل نصب خواهد شد، این روشی است که مایکروسافت که برای توسعه‌دهنده‌های خود پیشنهاد کرده است.

لينك رسمي جهت دریافت اين بستهها : <https://visualstudio.microsoft.com/downloads/>

## ساخت آیکون برنامه برای پلتفرم Windows

جهت ساخت آیکون برنامه تحت پلتفرم Windows ابتدا فایل تصویر مورد نظر خود را با کیفیت بالا ایجاد کنید و در قالب .ico آن را ذخیره نمایید.  
در صورتی که از qMake استفاده می‌کنید در فایل .pro دستور زیر را وارد کنید:

```
windows {
    RC_FILE = windows/AppIcon.ico
}
```

در فایل app.rc موجود در مسیر windows محتوای زیر را اضافه کنید:

```
#include <windows.h>

VS_VERSION_INFO      VERSIONINFO
FILEVERSION          0, 5, 0, 0
PRODUCTVERSION       0, 5, 0, 0
FILEFLAGSMASK        0x3fL
FILEFLAGS            0
FILEOS               VOS_NT_WINDOWS32
FILETYPE              VFT_APP
FILESUBTYPE           VFT2_UNKNOWN
BEGIN
    BLOCK   "VarFileInfo"
    BEGIN
        VALUE   "Translation", 0x409, 1200
    END
    BLOCK   "StringFileInfo"
    BEGIN
```

```

BLOCK      "040904b0"
BEGIN
    VALUE      "CompanyName",      "Genyleap\0"
    VALUE      "FileDescription",   "MyApplication\0"
    VALUE      "FileVersion",      "0.5.0-Beta\0"
    VALUE      "InternalName",     "MyApplication\0"
    VALUE      "LegalCopyright",   "Copyright 2017 by Genyleap\0"
    VALUE      "OriginalFilename", "MyApplication.exe\0"
    VALUE      "ProductName",      "MyApplication\0"
    VALUE      "ProductVersion",   "0.5.0-Beta\0"
END
END
ICON      DISCARDABLE      "windows/app.ico"

```

محتوای فوق اطلاعات مربوط به توسعه دهنده، نسخه و مشخصات دیگر برنامه را همراه فایل اجرایی exe درج می کند. که در انتهای آن نیز از دستور ICON جهت فراخوانی آیکون برنامه استفاده شده است.

### نکات مهم در رابطه با پیش نیازات در موارد خاص

اگر چه طبق شرایط توضیح داده شده برنامه به درستی و بدون هیچ خطای اجرا می شود، ولی در موارد خاص اگر شما از یک سری مازول ها و یا کتابخانه های دیگر C++ استفاده کنید بر اساس نیاز باید آن ها را شناسایی و در کنار پروژه فراهم نمایید.

مواردی مانند Web engine ها OpenGL، Direct X، Web engine و یا سیستم های یونیکد وجود دارد که برای پشتیبانی از این موارد باید با روش های انحصاری خود آن ها پروژه را پیکربندی نماییم.

برای مثال اگر در کدهای خود از سیستم یونیکد استفاده می کنید که شامل کتابخانه های ICU است، شما باید کتابخانه های مربوطه را در کنار برنامه فراهم سازید که هر کدام نسخه های متفاوتی را دارا می باشند که در جدول زیر نسخه های موجود و هماهنگ با Qt5 مشخص شده است که معمولاً آخرین نسخه های این کتابخانه در سایت رسمی آن موجود است.

نام فایل		
icudtXX.dll	icuinXX.dll	icuucXX.dll

فایل های مورد نیاز در سرور جنی لیپ برای عموم موجود است که طبق لینک های ذکر شده می توانید از آنها استفاده کنید، معمولاً پروژه هایی که نیازمند این فایل ها هستند هنگام اجرا پیغامی خواهند داد که حاوی نام و نسخه فایل مرتبط با کتابخانه ICU است و زمانی مورد نیاز هستند که کتابخانه Qt را بر پایه آن پیکربندی کرده باشید.

نکته : در صورتی که از کامپایلر **MinGW** استفاده می کنید فایل های فوق مورد نیاز خواهند بود : libstdc++-6.dll، libgcc\_s\_dw2-1.dll و libwinpthread-1.dll را می توانید از مسیر مربوط به کامپایلر MinGW کپی C:\Qt\Qt5.9.0-MinGW\5.9\mingw492\_32\bin و در کنار برنامه قرار دهید.

### پیکربندی و انتشار برنامه در پلتفرم مک (macOS)

یکی از مهمترین پلتفرم هایی که در توسعه محصولات بسیار مهم است پلتفرم اپل با نام macOS بر پایه یونیکس است. همانطور که می دانید شرکت اپل محصولات خود را طوری طراحی و منتشر کرده است که برای تولید و توسعه برنامه های مختلف بر روی آنها نیازمند سیستم عامل

مک و محیطهای توسعه انحصاری آن با نام Xcode هستیم. بر اساس سیاست‌های شرکت زبان‌های برنامه‌نویسی پیشفرض برای تولید و توسعه برنامه در پلتفرم‌های iOS و macOS Objective-C یا Swift هستند.

با توجه به ویژگی‌ها و قابلیت‌هایی که C++ نسبت به این زبان‌ها دارد جهت توسعه محصول توسط کیوت این امکان وجود دارد که محصولات خود را برای پلتفرم‌های مک و آی او اس توسعه و گسترش نماییم. اما قوانین و شرایطی حاکم بر این سکو هستند که باید آن‌ها را رعایت و طبق اصول معرفی شده و استاندارد اقدام به این کار کنیم که در ادامه به آن‌ها اشاره شده است.

### نکات قابل توجه برای این پلتفرم

بسیاری از پروژه‌ها چند سکویی می‌توانند به طور پایه توسط ویژگی‌های qmake پیکربندی شوند. با این حال که در برخی از پلتفرم‌ها این ویژگی مفید است و یا حتی در صورت لزوم ویژگی‌های خاصی را می‌توان فراهم آورد. بنابراین ابزار qmake بسیاری از این ویژگی‌ها را در اختیار ما قرار می‌دهد و ما می‌توانیم از طریف متغیرهای خاص که تنها بر روی پلتفرم مرتبط و مورد نظر موجود هستند به برخی از این ویژگی‌ها دسترسی داشته باشیم.

ویژگی‌هایی که مشخص شده‌اند برای پلتفرم‌های macOS، iOS، tvOS و watchOS قابل دسترس هستند و کیوت آن‌ها را در قالب چهارچوب پشتیبانی می‌کند.

نسخه‌ای که از qmake در بسته منبع باز عرضه شده است کمی متفاوت تر از نسخه عرضه شده در بسته باینری است که در آن ویژگی‌ها و خصوصیات متفاوتی فراهم شده است. به طور معمول بسته‌های منبع با استفاده از خصوصیات macx-g++ استفاده می‌کنند و بسته‌های باینری از خصوصیات macx-Xcode که هریک دارای ویژگی‌های پیکربندی شده خاص خود هستند.

کاربران هریک از بسته‌ها در صورتی که نیاز به در نظر نگرفتن این ویژگی‌ها باشند کافی است با استفاده از گزینه spec در qmake آن را اعمال کنند. برای مثال با استفاده از ابزار qmake جهت ایجاد یک بسته باینری برای ساخت فایل Makefile کافی است دستور زیر را اجرا کنیم:

```
qmake -spec macx-g++
```

ابزار qmake قادر است تا به طور خودکار قوانین ساخت برنامه را جهت اتصال در برابر چهارچوب استاندارد موجود در دایرکتوری macOS که در مسیر /Library/Frameworks/ است را فراهم کند. بنابراین می‌توانیم مسیر چهارچوب‌های استاندارد دیگری را طبق پیوند دهنده (لينک) به متغیر QMAKE\_LFLAGS موجود در پروژه تعريف و استفاده کنیم. جهت این کار کافی است طبق مثال زیر عمل کنید:

```
QMAKE_LFLAGS += -F/path/to/framework/directory/
```

چهارچوب توسط گزینه -framework و نام آن در متغیر LIBS پیوند داده شده است.

```
LIBS += -framework TheFramework
```

هر کتابخانه می‌تواند به عنوان یک چهارچوب به یک پروژه اضافه شده و در قالب مشخصی از آن استفاده شود. برای فراهم آوردن پروژه در قالب چهارچوب و استفاده از آن در دیگر پروژه‌ها باید از مقدار **LIB** در متغیر TEMPLATE و افزودن گزینه **lib\_bound** در متغیر CONFIG استفاده شود. این کار جهت ایجاد خروجی نهایی به عنوان یک کتابخانه ضروری است.

```
TEMPLATE = lib
CONFIG += lib_bundle
```

لازم است به این نکته توجه کنیم که اطلاعات مرتبط با کتابخانه با استفاده از متغیر **QMAKE\_BOUNDLE\_DATA** مشخص می‌شوند. این متغیر آیتم‌هایی را که قرار است همراه یک بسته نصب شوند را نگه می‌دارد و اغلب برای تعیین مجموعه‌ای از سرآیند فایل‌ها مورد استفاده قرار می‌گیرد که به طور زیر آورده شده است.

```
FRAMEWORK_HEADERS.version = Versions
FRAMEWORK_HEADERS.files = path/to/header_one.h path/to/header_two.h
FRAMEWORK_HEADERS.path = Headers
QMAKE_BOUNDLE_DATA += FRAMEWORK_HEADERS
```

شما با استفاده از متغیر **FRAMEWORK\_HEADERS** برای تعیین عناوین مورد نیاز توسط یک چهارچوب خاص را مشخص می‌کنید. اضافه کردن آن به متغیر **QMAKE\_BOUNDLE\_DATA** تصمین می‌کند که اطلاعات در مورد این سرآیند فایل به مجموعه‌ای از منابع که قرار است نصب شوند اضافه خواهد شد. همچنین نام چهارچوب و نسخه آن با متغیرهای **QMAKE\_FRAMEWORK\_BOUNDLE** و **QMAKE\_FRAMEWORK\_VERSION** مشخص شده‌اند.

## ساخت آیکون برنامه برای پلتفرم macOS

جهت ساخت آیکون برنامه تحت پلتفرم macOS ابتدا فایل تصویر مورد نظر خود را با کیفیت بالا ایجاد کنید و در قالب icns آن را ذخیره نمایید. برخی از ابزارها جهت تولید فایل آیکون با پسوند icns موجود هستند که نمونه آن توسط شرکت اپل با نام (iconutil) در دسترس است. در صورتی که از qMake استفاده می‌کنید در فایل pro. دستور زیر را وارد کنید:

```
macx {
    QMAKE_INFO_PLIST = macos/Info.plist
    ICON = macos/AppIcon.icns
}
```

در فایل Info.plist موجود در مسیر macos محتوای زیر را اضافه کنید:

```
<key>CFBundleIconFile</key>
<string>AppIcon.icns</string>
```

## نسخه‌های وابسته به macOS

تمامی نسخه‌های موجود در کیوت ۵ قابل ساخت در آخرین نسخه سیستم‌عامل اپل هستند که معمولاً قابلیت اجرا بر روی نسخه‌های پیشین را نیز دارند که توسط پیوند کننده ضعیفی صورت می‌گیرد که بهتر است بر اساس سیاست‌های اپل به روز رسانی ویژگی‌ها را در نظر داشته باشیم. بنابراین توسط متغیر MACOSX\_DEPLOYMENT\_TARGET می‌توان پروژه را برای نسخه‌های مختلفی از سیستم‌عامل مک تنظیم کرد که در زیر مثالی از پشتیبانی برای نسخه مورد نظر مشخص شده است:

```
QMAKE_MACOSX_DEPLOYMENT_TARGET = 10.12
```

این دستور با درج در فایل pro. مشخص می‌کند که برنامه بر روی نسخه ۱۰.۱۲ سیستم عامل macOS قابل اجرا خواهد بود.

```
win32 { ... }
android { ... }
ios { ... }
macx {
    QMAKE_MACOSX_DEPLOYMENT_TARGET = 10.3
}
```

## ابزار خروجی گرفتن (استقرار) در macOS

ابزار مربوطه را می‌توانید در مسیر QTDIR/bin/macdeployqt پیدا کنید. ابزار فوق به گونه‌ای طراحی شده است تا به طور خودکار روند ایجاد یک بسته نرم‌افزاری گسترش پذیر را که شامل کتابخانه‌های کیوت به صورت خصوصی هستند را فراهم کند. به طور کلی این ابزار امکان این را نیز فراهم می‌کند که پلاگین‌های مورد استفاده در پروژه همراه برنامه ساخته شوند مگر اینکه آن‌ها را توسط گزینه no-plugins غیرفعال سازید بنابراین نکات زیر نیز قابل توجه خواهد بود:

- پلاگین‌پلتفرم همیشه قابل ادغام با برنامه هستند.
- نسخه‌های مربوط به اشکال زدایی یا همان (Debug) بر روی خروجی نهایی مستقر نمی‌شوند.
- پلاگین‌های طراحی قابل استقرار نمی‌باشند.
- پلاگین‌های تصویر همیشه قابل استقرار بر روی پروژه نهایی هستند.
- پلاگین مخصوص پرینت قابل استقرار بر روی پروژه نهایی است.
- پلاگین‌های مخصوص پایگاه داده (SQL) همیشه در صورتی که پروژه از ماژول QSql استفاده کند مستقر خواهد شد.
- پلاگین‌های مخصوص اسکریپت در صورتی که ماژول QtScript در پروژه استفاده شده باشد همراه آن مستقر خواهد شد.
- پلاگین مهم SVG جهت نمایش فایل‌های icon و ... در صورت استفاده از ماژول QtSvg همراه پروژه مستقر خواهد شد.
- پلاگین‌هایی که در دسترس هستند همیشه قابل استقرار بر روی خروجی نهایی برنامه خواهند بود.
- همچنین دقت کنید که جهت افزودن کتابخانه نوع سوم (3<sup>rd</sup> party) در داخل بسته نرم‌افزاری خود باید آن را به صورت دستی در بسته مورد نظر بعد از ایجاد بسته نهایی ایجاد کنید.

## بنابراین ابزار macdeployqt گزینه‌های زیر را پشتیبانی می‌کند:

توضیحات	گزینه
دیباگ = 3, عادی = 2 خط/اخطر = 1, بدون خروجی = 0	-verbose=<0-3>
پرش از استقرار سازی پلاگین	-no-plugins
ایجاد یک بسته در قالب فایل dmg.	-dmg
اجرا نکردن 'strip' بر روی باینری	-no-strip

توضیحات	گزینه
استقرار با نسخه اشکال زدایی از چهارچوب و پلاگین	-use-debug-libs
امکان سازی برای اجرای برنامه در چهارچوب مستقر شده در مسیر مشخص	-executable=<path>
استقرار فایل‌های qml. برای اجرا در مسیر مشخص شده	-qmlDir=<path>

بر اساس استاندارد سکو و کیوت جهت ایجاد خروجی در یک پروژه ابزار مربوطه بسیار مفید و کارآمد خواهد بود لذا شما می‌توانید توسط آن به سادگی برنامه خود را در قالب یک بسته نرم‌افزاری همراه با تمامی پیش‌نیازات آن مستقر سازی نمایید. برای اینکار فرض کنید برنامه‌ای ساخته‌اید که از فناوری Qt Quick پشتیبانی می‌کند و دارای مازول کنترل‌x Quick Controls 2 است. مسلماً این برنامه باید در کنار خود کتابخانه‌ها و فایل‌های مورد نظر را باید مستقر سازی کند.

برای مثال ما برنامه‌ای ساخته‌ایم که در مسیر مورد نظر /Users/Compez/Desktop/MyAppRoot است. با دستور cd به مسیر فوق خواهیم

رفت:

```
cd /Users/Compez/Desktop/MyAppRoot
```

البته قرار است در این مسیر خروجی فایل بعد از کامپایل ایجاد شود که با غیرفعال‌سازی امکان Shadow Build این ممکن خواهد شد که فایل مربوطه در مسیر ریشه برنامه‌ایجاد شود. با فرض اینکه بعد از کامپایل فایل MyApplication.app در مسیر ذکر شده موجود باشد دستور زیر را در ترمینال وارد خواهیم کرد:

```
/Users/Compez/Qt/5.9/clang_64/bin/macdeployqt MyApplication.app -dmg
```

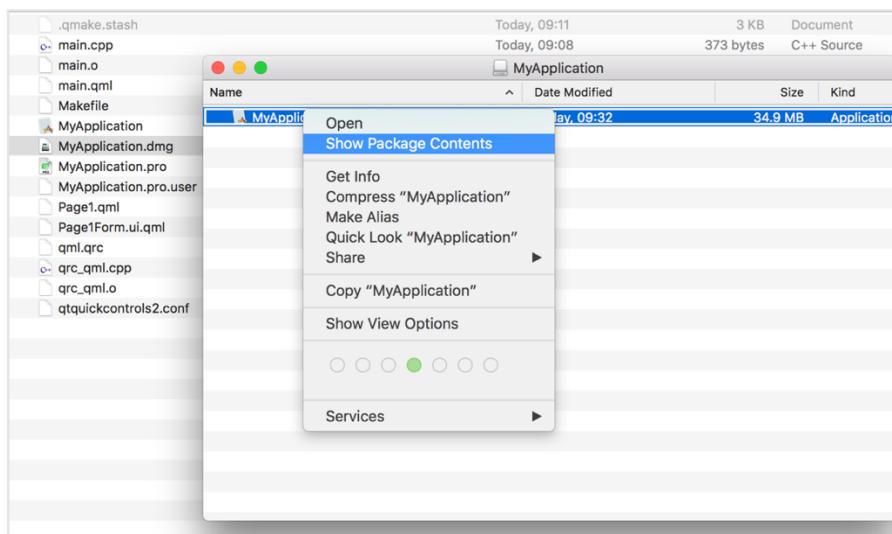
دقیق نیاز باشد با استفاده از گزینه‌های موجود در ابزار برنامه خود را مستقر سازید کافی است دستور ایجاد را به صورت زیر وارد کنید:

```
/Users/Compez/Qt/5.9/clang_64/bin/macdeployqt MyApplication.app -dmg -verbose=3 -no-plugins
```

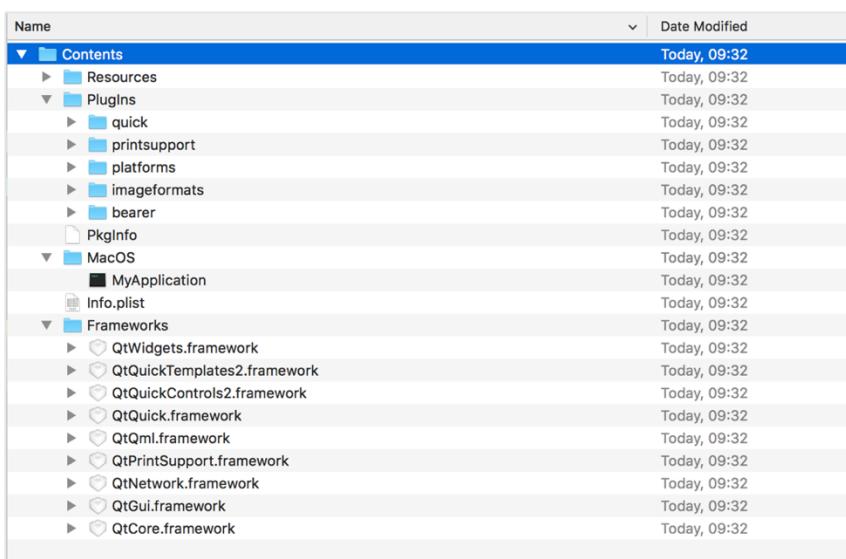
دستور فوق ابزار macdeployqt را از مسیر موجود فراخوانی کرده و سپس با مقدار MyApplication.app ترکیب و خروجی آن را در قالب dmg مخصوص پلتفرم macOS مستقر می‌سازد، بر اساس این دستور خروجی نتایج به صورت زیر خواهد بود:

```
Kambizs-MacBook-Pro:MyAppRoot Compez$ /Users/Compez/Qt/5.9/clang_64/bin/macdeployqt MyApplication.app -dmg
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquick2plugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2plugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2materialstyleplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2universalstyleplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libquicklayoutsplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquicktemplates2plugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libwindowplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2materialstyleplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2materialstyleplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2universalstyleplugin.dylib"
File exists, skip copy: "MyApplication.app/Contents/PlugIns/quick/libqtquickcontrols2universalstyleplugin.dylib"
Kambizs-MacBook-Pro:MyAppRoot Compez$
```

در نهایت خروجی ساخته شده در قالب `MyApplication.dmg` در مسیر فوق ایجاد خواهد شد. حال بعد از اجرای فایل `dmg` و نصب آن بر روی فایل `MyApplication.app` کلیک کرده و گزینه `Show Package Contents` را انتخاب کنید.



در ادامه خواهید دید که پلاگین‌ها و مازول‌های مربوطه در کنار هم به صورت یک بسته‌ایجاد و ساخته شده‌اند و نیازی برای نگرانی از اجرای برنامه به صورت مستقل از کیوت نخواهید داشت.



در ادامه خواهید دید که پلاگین‌ها و مازول‌های مربوطه در کنار هم به صورت یک بسته‌ایجاد و ساخته شده‌اند و نیازی برای نگرانی از اجرای برنامه به صورت مستقل از کیوت نخواهید داشت در نهایت تنها کافی است فایل `dmg` را در سایت یا پکیج‌های مورد نظر خود قرار دهید تا کاربران مک آن را دریافت و از برنامه شما بهره‌مند شوند.

## پیکربندی و انتشار برنامه در پلتفرم لینوکس (Linux)

لینوکس، به عنوان یک سیستم‌عامل پرکاربرد یکی از پلتفرم‌هایی است که افراد نه چندان مبتدی از آن استفاده می‌کنند. لذا محیطی بسیار جذاب و دوست داشتنی برای برنامه‌نویسان نیز محسوب می‌شود. اما بر خلاف دو سیستم‌عامل قبلی از ابزار استقرار کننده برنامه برخوردار نیست و باید به روش کمی حرفه‌ای تر برنامه خود را مستقر سازی کنیم.

برای اینکار بعد از کامپایل برنامه برای اجرا خارج از محیط Qt به مسیر موجود فایل اجرایی رفته و در ترمینال دستور زیر را وارد کنید:

```
./MyApplication
```

برنامه بعد از اجرای دستور فوق اجرا خواهد شد. اما برای مستقر سازی آن باید متوجه شویم که چه کتابخانه‌هایی تحت برنامه ما فراخوانی می‌شوند که برای اینکار کافی است در مسیر فوق دستور زیر را وارد کنید:

```
ldd ./MyApplication
```

تمامی کتابخانه‌هایی که در برنامه مورد استفاده قرار می‌گیرند در خروجی ترمینال قابل مشاهده خواهند بود که برای کپی فایل‌های مورد نیاز به صورت ساختار زیر عمل خواهیم کرد.

نام فایل	کامپوننت
MyApplication	فایل اجرایی
MyApplication.sh	اسکریپت اجرا کنندهٔ فایل اجرایی
plugins\libpnp_basictools.so	ابزار پایهٔ پلاگین‌ها
plugins\libpnp_extrafilters.so	ابزار فیلترینگ پلاگین‌ها
platforms\libqxcb.so	فایل پیش نیاز پلتفرم کیوت
libQt5Core.so.5	فایل هستهٔ کیوت
libQt5Gui.so.5	فایل گرافیکی GUI کیوت
libQt5Qml.so.5	فایل مربوط بهٔ ماژول QML در کیوت
libQt5{ماژول}.so.5	تمامی ماژول‌هایی که در پروژه استفاده شده را کپی کنید

در پروژه فوق ما از فناوری Quick Qt نیز استفاده کرده‌ایم بنابراین ماژول‌های مرتبط با شبکه و همچنین QML را باید در کنار برنامه اصلی قرار دهید که برای این کار ابتدا وارد مسیر نصب شده Qt با پوشه gcc شویم. فایل‌های ذکر شده را کپی کرده و در کنار فایل اجرایی یعنی MyApplication قرار دهید.

جهت ساخت فایل اجرایی قابل اجرا در سکوی لینوکس وارد مسیر کامپایل شده خروجی پروژه شده و یک فایل با نام برنامه خود و با پسوند .sh جهت ساخت فایل اجرایی قابل اجرا در سکوی لینوکس وارد مسیر کامپایل شده خروجی پروژه شده و یک فایل با نام برنامه خود و با پسوند .sh بسازید **MyApplication.sh** و سپس محتوای آن را به صورت زیر وارد نمایید.

دقت کنید که نام فایل باید دقیقاً نام برنامه شما باشد.

```
#!/bin/sh
MyApplication=`basename $0` | sed s,\.sh$,, `

dirname=`dirname $0`
tmp="${dirname##?}""

if [ "${dirname%$tmp}" != "/" ]; then
```

```

dirname=$PWD/$dirname
fi
LD_LIBRARY_PATH=$dirname
export LD_LIBRARY_PATH
$dirname/$MyApplication "$@"

```

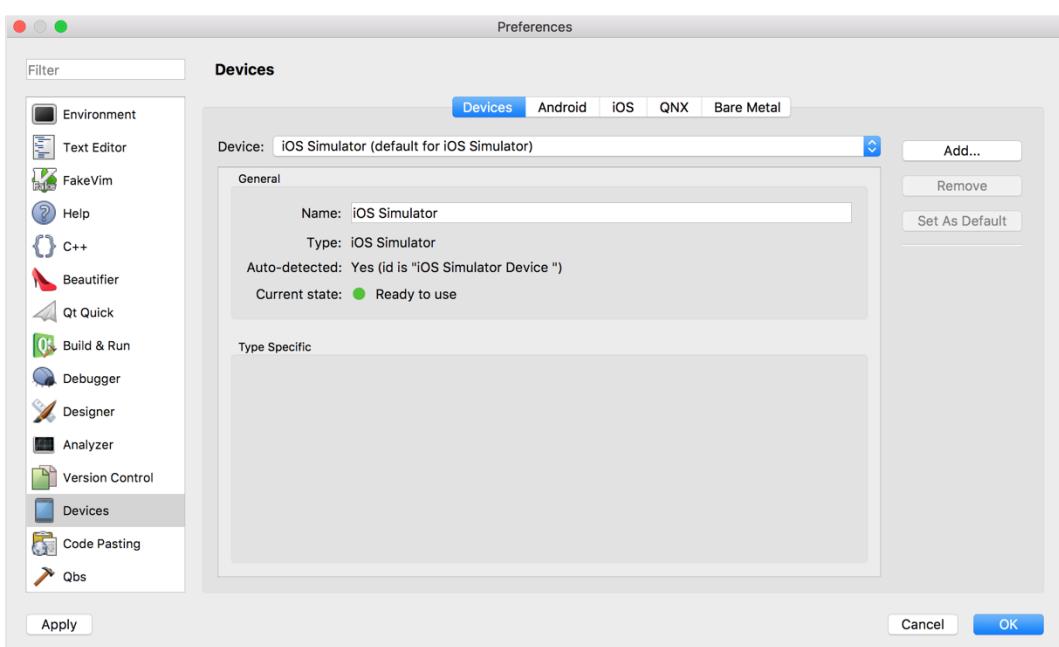
ترمینال را باز کرده و دستور زیر را اجرا کنید.

```
chmod +x MyApplication.sh
```

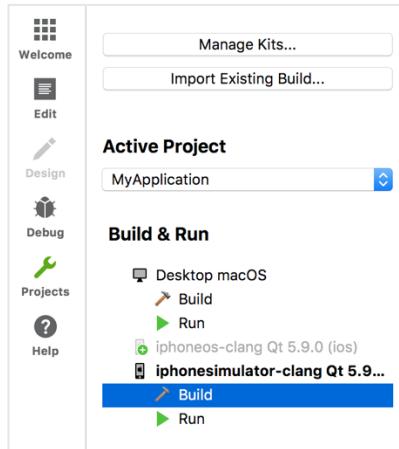
برنامه شما در قالب یک فایل اجرای با پسوند sh. مستقر شده است که می‌توانید با کلیک بر روی آن برنامه را اجرا کنید.

### پیکربندی و انتشار برنامه در پلتفرم گوشی‌ها و تبلت‌های هوشمند iPad و iPhone و (سیستم‌عامل iOS)

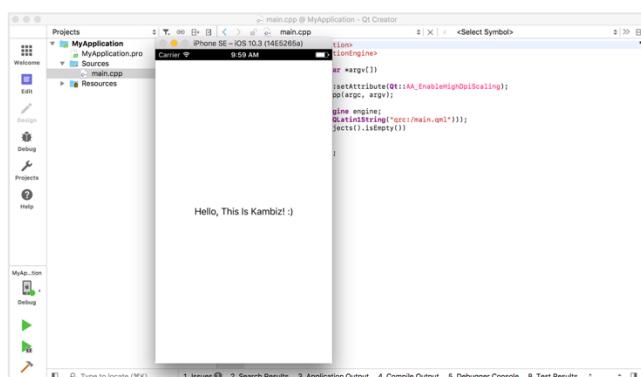
جدابترین بخش کیوت در توسعه محصولات بر روی دستگاه‌های هوشمند موبایل این است که قابلیت پشتیبانی از انواع محصولات شرکت اپل فراهم شده است که صد البته تمامی این قابلیت‌ها را مدیون ساختار اصلی C++ هستیم که به طور چند منظوره و مستقل از پلتفرم کاربرد دارد. بنابراین به قسمت Devices رفته و در زبانه Qt Creator->Preferences از وضعیت سیستم شبیه ساز مطلع خواهیم شد.



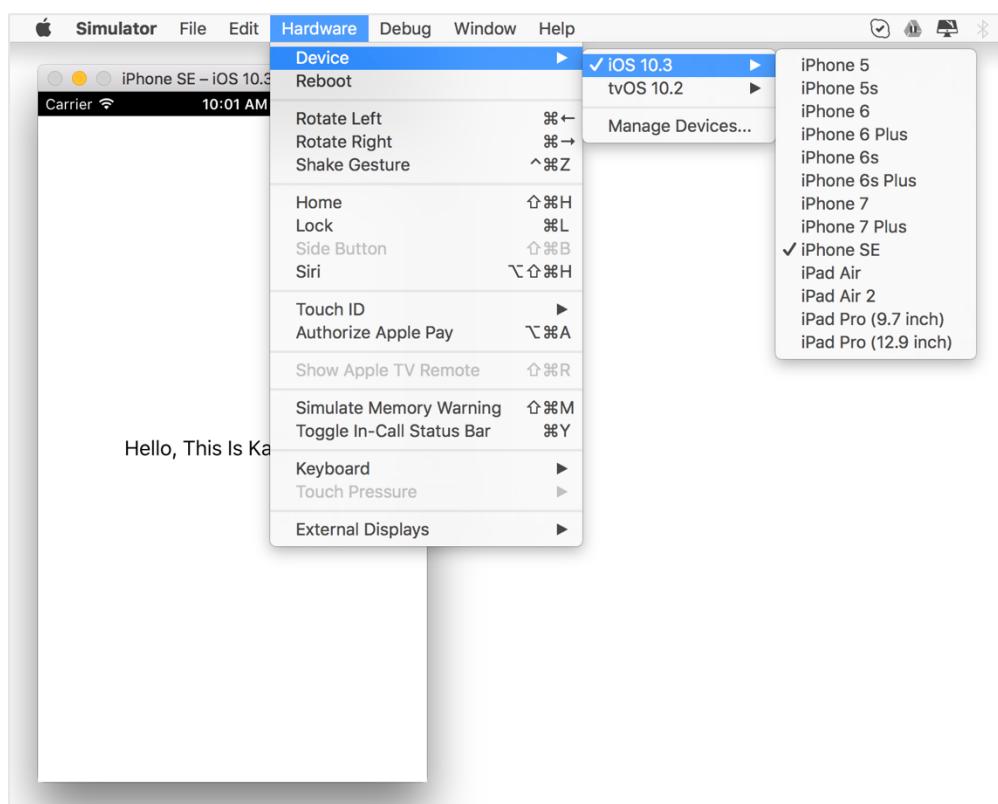
به طور کلی Qt امکان این را فراهم می‌سازد که بتوانیم برنامه‌ها را بر روی پلتفرم‌های مربوط به iPod، iPad و iPhone و iPod مستقر سازی کنیم. البته توجه داشته باشید که برای اجرای برنامه بر روی شبیه‌ساز iOS شما بدون دغدغه خاصی تنها با داشتن Apple Developer ID می‌توانید برنامه را بر روی شبیه ساز اجرا و مورد آزمایش قرار دهید. برای این کار کافی است وارد قسمت Project شده و از قسمت Build & Run گزینه iphonesimulator-clang Qt 5.9.0 را انتخاب کنید.



در نهایت بعد از کامپایل و اجرا برنامه بر روی شبیه‌ساز iOS به صورت زیر اجرا خواهد شد:

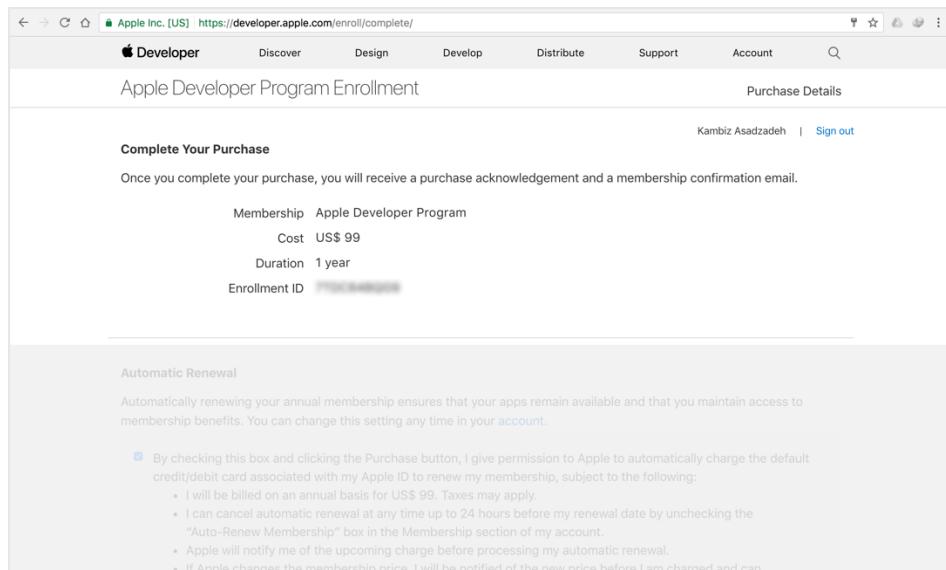


یک نکتهٔ کاربردی که شاید نیاز باشد برنامه خود را بر روی نسخه‌های مختلفی از محصولات اپل اجرا و مورد آزمایش قرار دهید، در این صورت می‌توانید با مراجعه به قسمت Simulator->Hardware->Device پلتفرم مورد نظر خود را انتخاب و برنامه را مجدداً کامپایل و اجرا کنید.

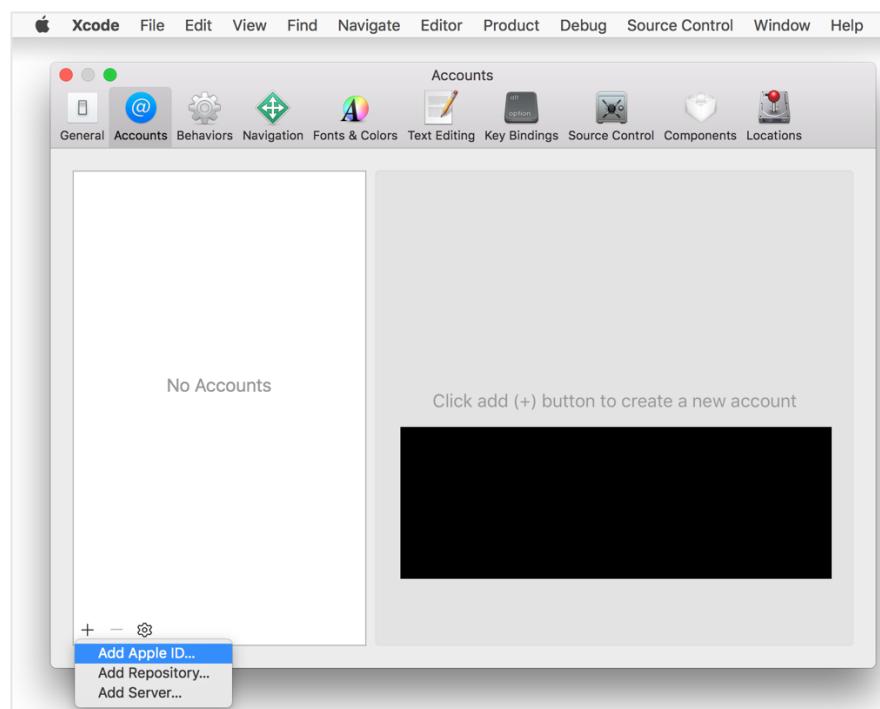


حال در نظر داریم برنامه مورد نظر خود را بر روی دستگاه واقعی آیفون اجرا کنیم. قبل از هرچیز توجه داشته باشید که باید حساب توسعه‌دهندگی خود را در سایت اپل ایجاد کنید. حساب کاربری اپل می‌تواند همان حساب توسعه‌دهندگی باشد تنها کافی است اکانت خود را برای دسترسی به

مجوزهای انتشار بر روی Apple Store و اجرا بر روی دستگاههای مختلف iPhone و غیره ثبت‌نام حساب توسعه‌دهندگی خود را از طریق لینک <https://developer.apple.com/programs/enroll/> عملی و نهایی سازی کنید. در نهایت بعد از وارد شدن به صفحه حساب کاربری اپل برای اعطای مجوز توسعه‌دهندگی مبلغ ۹۹ دلار در سال را با شناسه رسمی برای شما اعمال خواهد کرد.



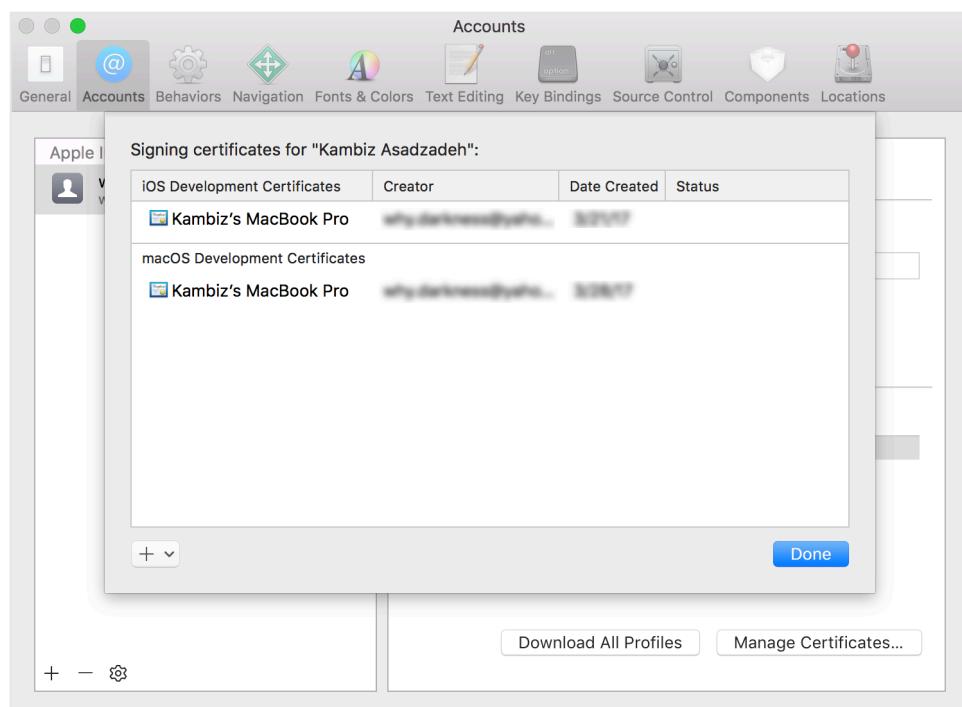
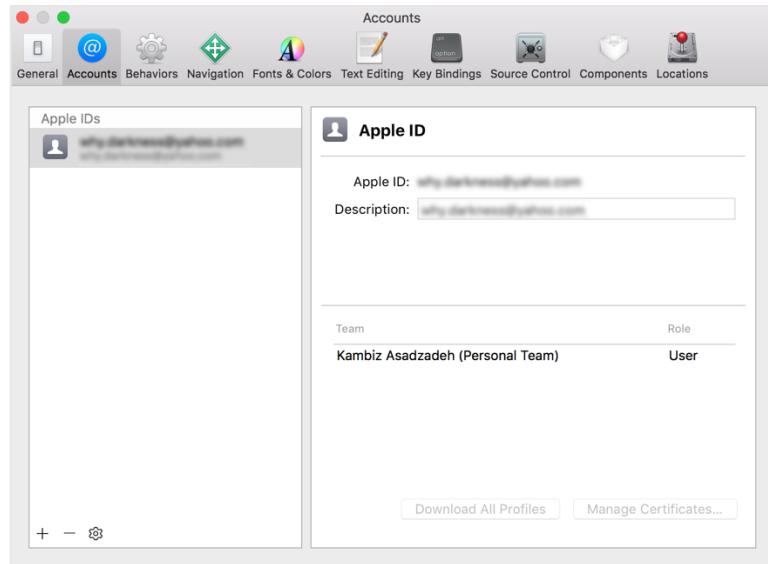
در صفحه فوق مبلغ عضویت و همچنین اعتبار آن مشخص شده است که Enrollment ID نیز نگهدارنده شناسه انحصاری پرداخت عضویت برای توسعه‌دهنده است. با فرض اینکه اکانت توسعه‌خود را ساخته باشیم کافی است وارد برنامه Xcode شده و گزینه Preferences را انتخاب کنیم، سپس وارد زبانه Add Apple ID کلیک کنیم.



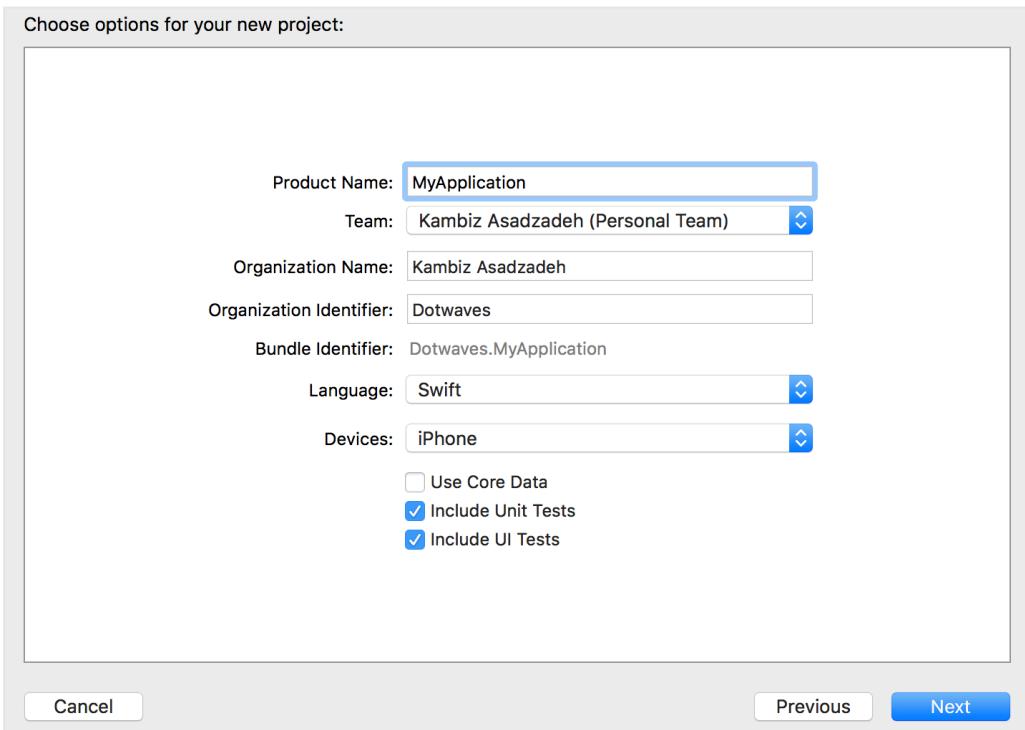
در مرحله بعد اطلاعات حساب کاربری مربوط به اپل را وارد خواهیم کرد.



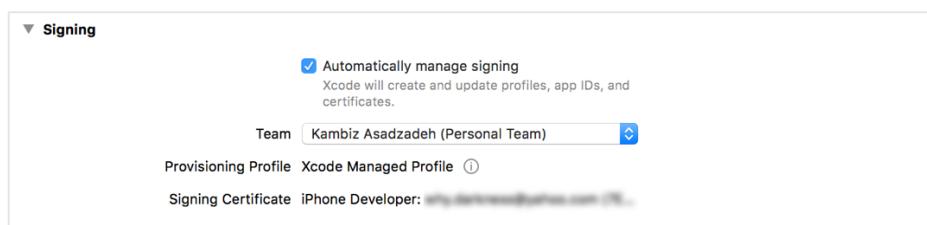
در نهایت می‌توانیم حساب کاربری خود را تایید و مورد استفاده قرار دهیم. بر روی اکانت خود دوبار کلیک کرده و مشخصات آن را بررسی کنید. در صورتی که مورد تایید قرار نگرفته باشد امکان افزودن گواهی‌نامه در لیست ممکن نخواهد بود. بنابر این در ادامه اطلاعات زیر را خواهیم داشت.



حال محیط Xcode را باز کنید و پروژه‌ای را در آن ایجاد کنید. این کار برای این است که ما اطلاعات حساب کاربری توسعه‌دهندگی را باید برای بار اول در محیط Xcode وارد کنیم تا گواهی‌نامه مورد تایید قرار گیرد، توجه کنید که حتماً نام پروژه همان نامی باشد که قرار است برای پروژه خودتان در Qt انتخاب کنید. در اینجا QApplication همان نامی است که قرار است برای آن گواهی‌نامه صادر شود. در غیر اینصورت موقع کامپایل خطای "no provisioning profiles found" را دریافت خواهید کرد.



در بخش بعدی صبر کنید تا اطلاعات شما تایید شود. سپس از منوی Product گزینه Build For Profiling را انتخاب و در این حالت فایل info.plist برای MyApplication ایجاد خواهد شد.



در این مرحله برای اینکه بتوانیم کیوت را توسط این شناسه توسعه دهنده هماهنگ سازیم در نسخه Qt Creator 4.3.0 به بعد امکان آن ایجاد شده است که با مراجعه به بخش Projects و سپس iOS Setting در زبانه Build & Run مقدار Development Team خود را بر روی حساب کاربری توسعه دهنده قرار دهیم تا معادل سازی برای موارزیر صورت گیرد:

```
QMAKE_XCODE_CODE_SIGN_IDENTITY = "iPhone Developer"
MY_DEVELOPMENT_TEAM.name = حساب توسعه شما (Email)
MY_DEVELOPMENT_TEAM.value = (RANDOM CHAR)
QMAKE_MAC_XCODE_SETTINGS += حساب توسعه شما (Email)
```

توجه داشته باشید که متغیرهای فوق در نسخه های قبل از ۴.۳ محیط توسعه Qt Creator حتما باید وارد می شدند اما در نسخه ۴.۳ به بعد نیازی برای باز نویسی آنها نیست زیرا این کار به طور خودکار با انتخاب Development Team در بخش Ios Setting برای صورت می گیرد.

خروجی دستور به صورت خودکار در کنسول محیط توسعه بعد از جرای دستور qMake ایجاد خواهد شد.

```

MyApplication.pro

-spec macx-ios-clang

CONFIG+=iphoneos

CONFIG+=device

CONFIG+=qml_debug

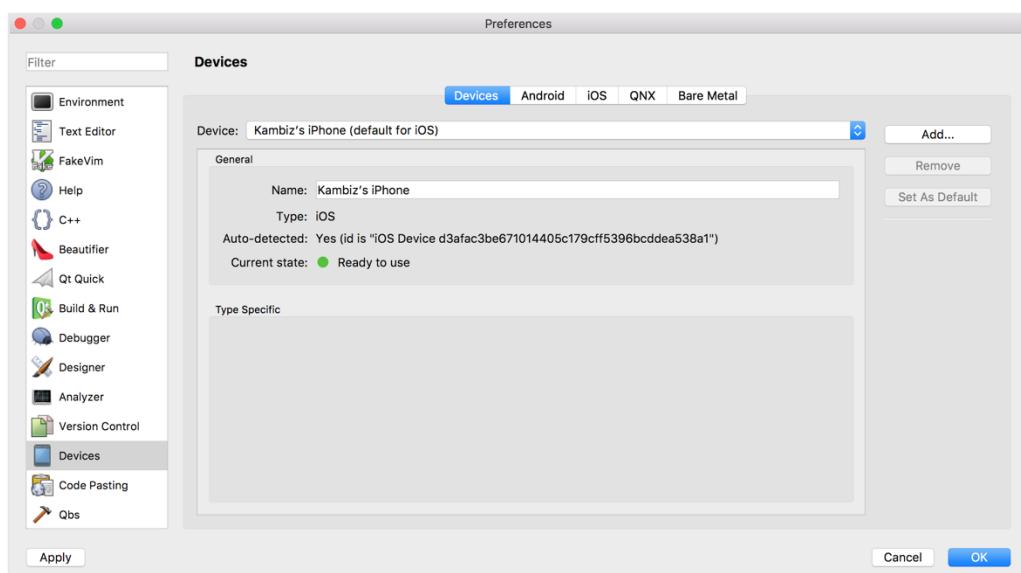
-after

QMAKE_MAC_XCODE_SETTINGS+=qteam

qteam.name=DEVELOPMENT_TEAM qteam.value=YourID

```

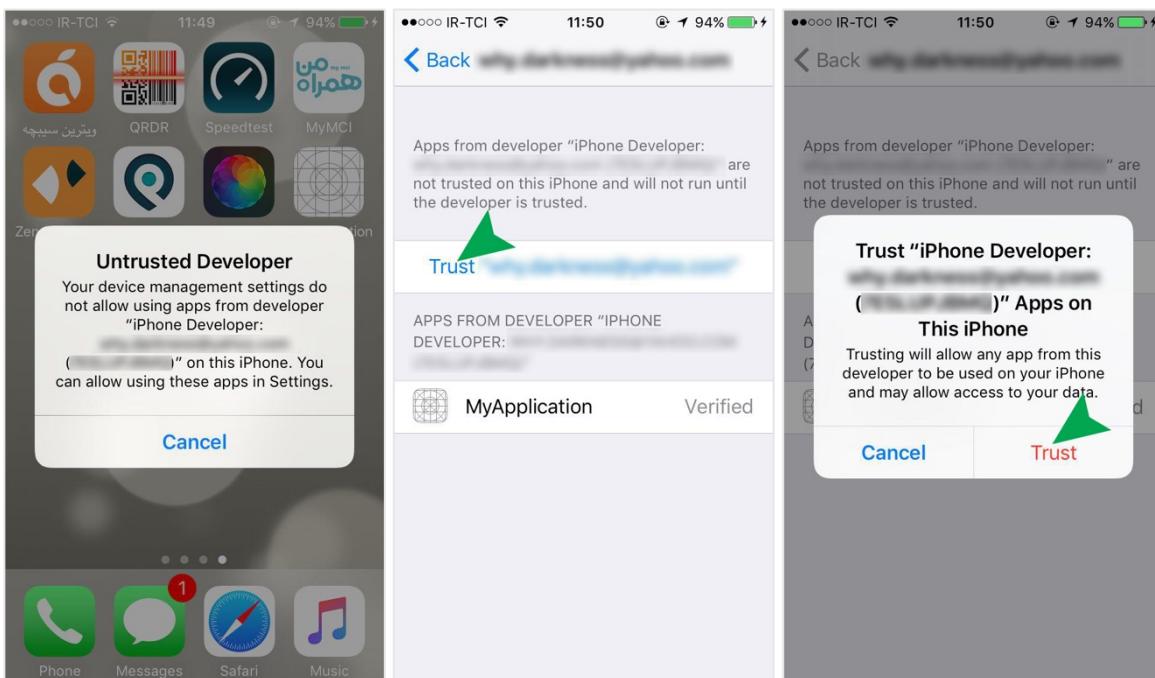
به منوی Qt Creator Preferences در Devices رفته و به زبانه مراجعه کنید. در این صورت باید دستگاه واقعی iPhone را به مکبوك خود متصل کرده باشید تا وضعیت آن به رنگ سبز و Ready to use تغییر کند.



حال پروژه را که متصل به دستگاه است کامپایل و اجرا کنید. برنامه بر روی دستگاه شما کامپایل و نصب شده است. اگر آن را اجرا کنید مسلماً پیغام زیر را دریافت خواهید کرد که معمولاً در صورتی که حساب کاربری شما شارژ نشده باشد مجبور خواهید بود به طور دستی آن را نصب و مورد تایید قرار دهید.



جهت ادامه به قسمت General در iOS Device Management رفته وارد شوید. در این بخش گزینه‌ای که در زبانه DEVELOPER APP موجود است را انتخاب کنید. در ادامه جهت تایید گزینه Trust را انتخاب کنید تا در ادامه آن را تایید نمایید.



در نهایت برنامه شما آماده اجرا و آزمایش بر روی دستگاه واقعی iOS خواهد بود.



در نهایت برنامه شما آماده اجرا و آزمایش بر روی دستگاه واقعی iOS خواهد بود. اما توجه داشته باشید که با این شرایط شما نمی‌توانید به هیچ عنوان برنامه خود را بر روی App Store قرار دهید. چرا که علاوه بر شارژ حساب کاربری نیاز خواهید داشت تا تنظیمات و قوانین استاندارد اپل را برای انتشار محصول بر روی فروشگاه آن رعایت کنید که در ادامه به آن‌ها اشاره شده است.

به طور کلی توسط Qt Creator شما می‌توانید برنامه خود را بر روی پلتفرم macOS جهت تولید، توسعه، اجرا و اشکال زدایی برنامه iOS انجام دهید. بنابراین ابزار Xcode اپل ابزار خود را با qmake ترکیب می‌کند تا تمامی پیش‌نیازات مورد نظر محصول در اختیار شما قرار گیرد چرا که کیوت به تنهایی تمامی ابزارهای خاص محیط اپل را فراهم نمی‌کند برای همین ممکن است لازم باشد بعضی اوقات به طور مستقیم از Xcode جهت ارائه برخی از ابزارهای ساخت و ساز استفاده کنیم که این کار را کیوت به طور خودکار انجام می‌دهد.

بررسی برنامه قبل از ارسال بر روی Apple Store برای اینکه درست اجرا شود و مشکلاتی نداشته باشد بسیار مهم است.

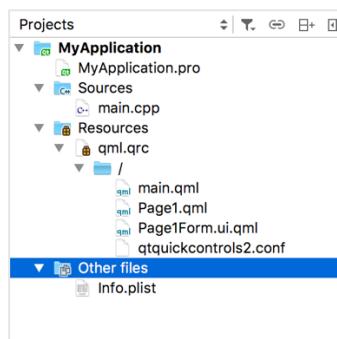
اطلاعات لیستی از اموال که به صورت فایل هستند در فایل Info.plist برای iOS و macOS جهت پیکربندی یک برنامه بسته بندی شده ذخیره می‌شود که گزینه‌های آن به صورت زیر آمده است:

- نام نمایشی برنامه و شناسه انحصاری آن
- قابلیت‌های مورد نیاز دستگاه
- پشتیبانی جهت‌گیری برای رابط کاربری
- راهاندازی و اجرای آیکون‌ها و تصاویر

زمانی که qmake اجرا شود یک فایل Info.plist با مقدار پیشفرض تولید خواهد شد. برای اینکه جلوگیری کنید از تولید مجدد این فایل با اطلاعات پیشفرض می‌توانید اطلاعات سفارشی خود را توسط متغیر QMAKE\_INFO\_PLIST برای پروژه خود تعریف کنید. بنابراین کد زیر را در فایل pro خواهیم داشت:

```
ios { QMAKE_INFO_PLIST = Info.plist }
```

بعد از افزودن آن و اجرای qmake این فایل به پروژه اصلی اضافه خواهد شد.

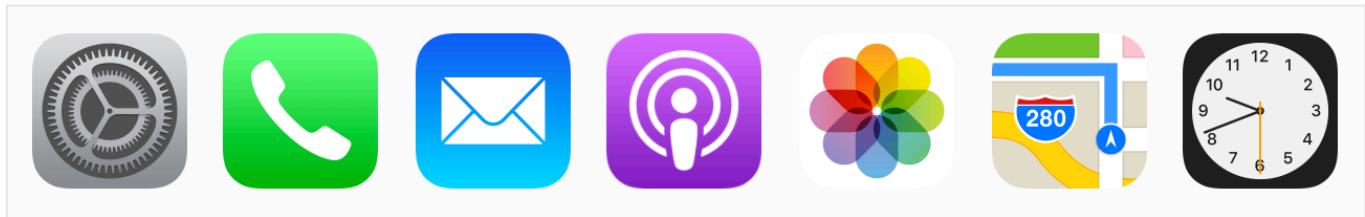


جهت تعریف فایلهایی که نمی‌توانند همراه با منابع کیوت مستقر شوند از متغیر QMAKE\_BUNDLE\_DATA استفاده خواهیم کرد که این متغیر روشی را فراهم می‌کند تا مجموعه‌ای از فایلهایی که قرار است در بسته نرم‌افزاری کپی شوند را فراهم می‌کند.

```
ios {  
    QMAKE_INFO_PLIST = ios/Info.plist  
    fontFiles.files = fonts/myFont.ttf  
    fontFiles.path = fonts  
    QMAKE_BUNDLE_DATA += fontFiles  
}
```

جهت تعریف فایلهایی که نمی‌توانند همراه با منابع کیوت مستقر شوند از متغیر QMAKE\_BUNDLE\_DATA استفاده خواهیم کرد که این متغیر روشی را فراهم می‌کند تا مجموعه‌ای از فایلهایی که قرار است در بسته نرم‌افزاری کپی شوند را فراهم می‌کند. این قابلیت زمانی مورد استفاده قرار می‌گیرد که قرار باشد فایلهایی را برای پروژه در نظر بگیریم. بنابراین با توجه به اینکه ایکون برنامه یکی از مهمترین مواردی است که به آن توجه می‌کند جهت ساخت و تعریف آیکون استاندارد برای iOS باید به صورت زیر عمل کنیم.

## آیکون برنامه در iOS و iPad برای iPhone



استانداردی که اپل برای آیکون‌ها تعریف کرده است به صورت زیر هستند که برخی از آن‌ها را که بیشترین کاربرد را بر روی دستگاه‌های iPad و iPhone دارند مشخص شده است:

- AppIcon29x29.png: 29 x 29
- AppIcon29x29@2x.png: 58 x 58
- AppIcon29x29@2x~ipad.png: 58 x 58
- AppIcon29x29~ipad.png: 29 x 29
- AppIcon40x40@2x.png: 80 x 80
- AppIcon40x40@2x~ipad.png: 80 x 80
- AppIcon40x40~ipad.png: 40 x 40
- AppIcon50x50@2x~ipad.png: 100 x 100
- AppIcon50x50~ipad.png: 50 x 50
- AppIcon57x57.png: 57 x 57
- AppIcon57x57@2x.png: 114 x 114
- AppIcon60x60@2x.png: 120 x 120
- AppIcon72x72@2x~ipad.png: 144 x 144
- AppIcon72x72~ipad.png: 72 x 72
- AppIcon76x76@2x~ipad.png: 152 x 152
- AppIcon76x76@2x~ipad.png: 76 x 76

بر اساس نیاز آیکون‌های مورد نظر را بعد از طراحی با ابعاد و بزرگ‌نمایی مورد نظر در پوشه‌ای با نام icons در زیر ios ایجاد خواهیم کرد. در صورتی که فایل Info.plist را ایجاد کرده باشید در این صورت وارد پوشه ios سپس Other files در نهایت فایل Info.plist را باز کنید.

در این فایل باید کلید و رشته‌های مرتبط با آیکون‌های مربوطه را که در مسیر ...\*.png ios/icons قرار گرفته اند را تعریف کنیم. وارد فایل شده و کد زیر را که مرتبط با دستورات تنظیم آیکون است وارد کنید:

```
<key>CFBundleIcons</key>
<dict>
<key>CFBundlePrimaryIcon</key>
```

```

<dict>
<key>CFBundleIconFiles</key>
<array>
<string>AppIcon-29x29@1x.png</string>
<string>AppIcon-29x29@2x.png</string>
<string>AppIcon-40x40@1x.png</string>
<string>AppIcon-40x40@2x.png</string>
<string>AppIcon-57x57@1x.png</string>
<string>AppIcon-57x57@2x.png</string>
<string>AppIcon-60x60@1x.png</string>
<string>AppIcon-60x60@2x.png</string>
<string>AppIcon-72x72@1x.png</string>
<string>AppIcon-72x72@2x.png</string>
<string>AppIcon-76x76@1x.png</string>
<string>AppIcon-76x76@2x.png</string>
</array>
</dict>
</dict>

```

کلیدهای **CFBundlePrimaryIcon** و **CFBundleIcons** مشخص می‌کنند که بسته‌ایکن‌ها برای برنامه برسی شود سپس توسط کلید **CFBundleIconFiles** آرایه‌ای را با ورودی‌های نوع رشتہ‌ای جهت نام فایل همراه با پسوند در بسته معرفی می‌کند. دقت کنید در صورتی که نیاز باشد نسخه‌های مرتبط با iPhone و iPad را جدا کنید کافی است یک کلید مخصوص iPad ایجاد کنید (**CFBundleIcons-ipad**) و برای نهایی سازی توسط دستورات کیوت وارد فایل pro.pbxproj مربوط به آن را تعریف کنید:

```

ios {
    QMAKE_INFO_PLIST = ios/Info.plist
    iosIcon.files = ios/icons/AppIcon29x29@1x.png \
                    ios/icons/AppIcon29x29@2x.png \
                    ios/icons/AppIcon40x40@1x.png \
                    ios/icons/AppIcon40x40@2x.png \
                    ios/icons/AppIcon57x57@1x.png \
                    ios/icons/AppIcon57x57@2x.png \
                    ios/icons/AppIcon60x60@1x.png \
                    ios/icons/AppIcon60x60@2x.png \
                    ios/icons/AppIcon72x72@1x.png \
                    ios/icons/AppIcon72x72@2x.png \
                    ios/icons/AppIcon72x72@2x.png \
                    ios/icons/AppIcon76x76@2x.png

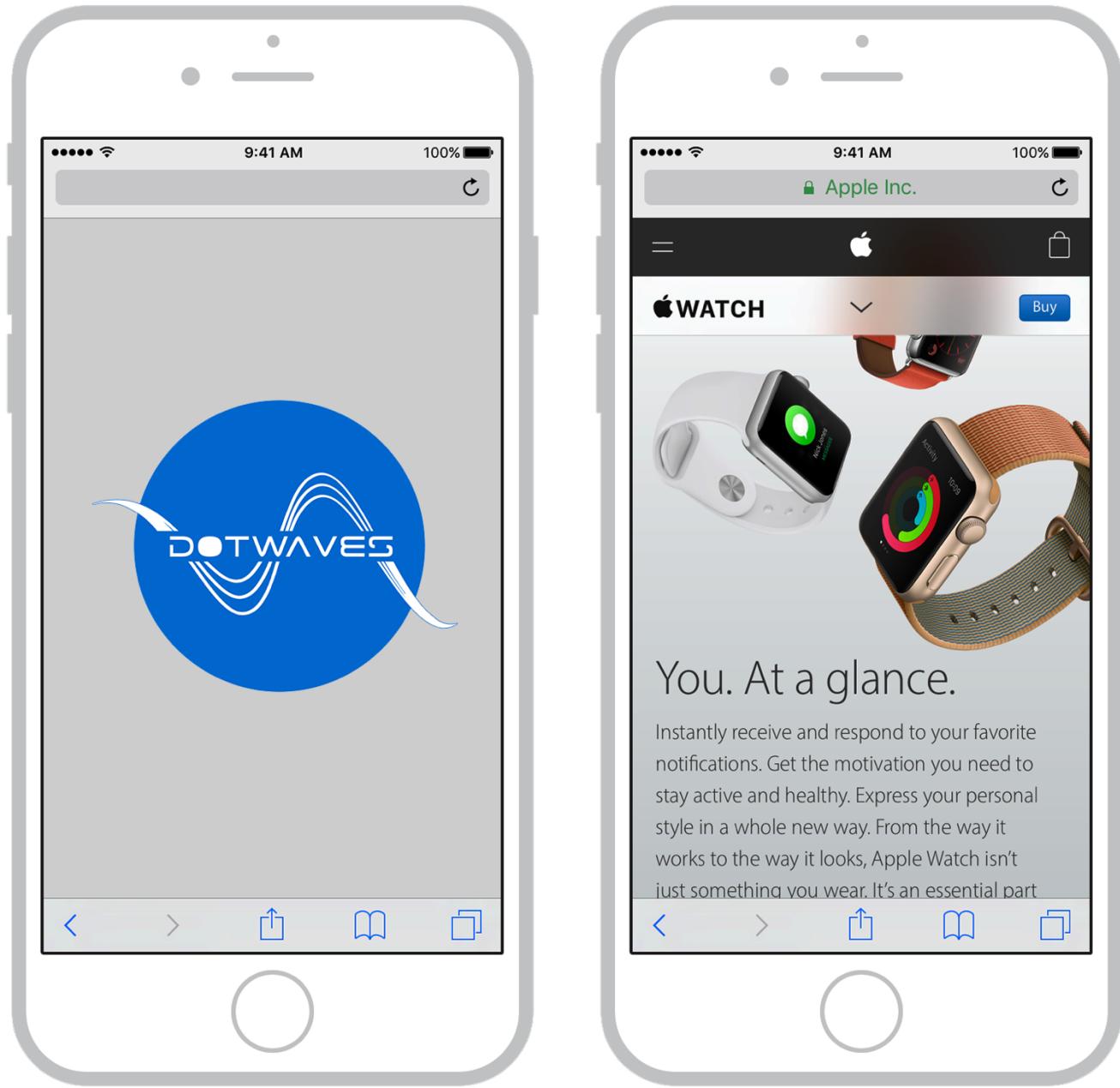
    QMAKE_BUNDLE_DATA += iosIcon
}

```

توجه داشته باشید که استانداردهای اپل بسیار حساس به این موضوع هستند، اگر آیکون شما بر اساس استاندارد تعریف شده در سایت اپل نباشد غیر ممکن است که برنامه شما مورد قبول قرار گیرد. کد فوق با مسیردهی به فایل‌های مربوط به‌ایکون‌های مخصوص iOS و در نهایت ادغام آن در بسته نرم‌افزاری یک آیکون برای برنامه در نظر می‌گیرد که بعد از کامپایل و نصب بر روی iPhone یا iPad قابل مشاهده خواهد بود.

## تصویر اجرایی قبل از بارگیری برنامه

علاوه بر آیکون برنامه تصویر اجرای اولیه در اپلیکیشن یکی از مواردی است که در هر پروژه ممکن است طراحی انحصاری داشته باشد که قبل از اجرا شدن برنامه قرار است نمایش داده شود. این تصویر قبل از بارگیری کامل برنامه سریع نمایش داده می‌شود و سپس مخفی خواهد شد.



این قابلیت بر اساس استاندارد زیر معرفی می‌شود:

LaunchImage.png:	320 x 480
LaunchImage@2x.png:	640 x 960
LaunchImage-568h@2x.png:	640 x 1136
LaunchImage-Landscape.png:	1024 x 748
LaunchImage-Landscape@2x.png:	2048 x 1496
LaunchImage-Portrait.png:	768 x 1004
LaunchImage-Portrait@2x.png:	1536 x 2008

تصاویر مربوط به Landscape باید طوری طراحی شوند که موقع گرفتن گوشی یا تبلت به صورت پهنا عمل کند و تصویر از لحاظ UI و UX مطابق با استانداردها باشد. برای ایجاد این قابلیت باید در فایل Info.plist دستورات لازم را وارد کنید. بنابراین استاندارد رسمی آن در زیر آمده است:

چرخش پهنا عریض	چرخش استاندارد	دستگاه
1920px x 1080px	1080px x 1920px	<b>iPhone 7 Plus, iPhone 6s Plus</b>
1334px x 750px	750px x 1334px	<b>iPhone 7, iPhone 6s</b>
1136px x 640px	640px x 1136px	<b>iPhone SE</b>

2732px × 2048px	2048px × 2732px	<b>12.9-inch iPad Pro</b>
2048px × 1536px	1536px × 2048px	<b>9.7-inch iPad Pro, iPad Air 2, iPad mini 4, iPad mini 2</b>

در حالت عادی دستور مرتبط به فایل Info.plist به صورت زیر خواهد بود:

```
<key>UILaunchImageFile</key>
<string>LaunchImage</string>
```

در ادامه کد دستوری برای فایل pro.plist به روش زیر است:

```
app_launch_images.files = $$files($$PWD/ios/icons/LaunchImage.png)
QMAKE_BUNDLE_DATA += app_launch_images
```

اگر نیاز باشد این بخش سفارشی سازی شود کافی است دستورات فوق را بر اساس نوع چرخش دستگاه اعمال کنید که در زیر آمده است:

```
<key>UILaunchImages</key>
<array>
  <dict>
    <key>UILaunchImageMinimumOSVersion</key>
    <string>7.0</string>
    <key>UILaunchImageName</key>
    <string>LaunchImage-iOS7</string>
    <key>UILaunchImageOrientation</key>
    <string>Portrait</string>
    <key>UILaunchImageSize</key>
    <string>{320, 568}</string>
  </dict>
  <dict>
    <key>UILaunchImageMinimumOSVersion</key>
    <string>7.0</string>
    <key>UILaunchImageName</key>
    <string>LaunchImage-iOS7</string>
    <key>UILaunchImageOrientation</key>
    <string>Portrait</string>
    <key>UILaunchImageSize</key>
    <string>{320, 480}</string>
  </dict>
</array>
<key>UILaunchImages~ipad</key>
<array>
  <dict>
    <key>UILaunchImageMinimumOSVersion</key>
    <string>7.0</string>
    <key>UILaunchImageName</key>
    <string>LaunchImage-iOS7-Landscape</string>
    <key>UILaunchImageOrientation</key>
    <string>Landscape</string>
    <key>UILaunchImageSize</key>
    <string>{768, 1024}</string>
  </dict>
  <dict>
    <key>UILaunchImageMinimumOSVersion</key>
    <string>7.0</string>
    <key>UILaunchImageName</key>
    <string>LaunchImage-iOS7-Portrait</string>
    <key>UILaunchImageOrientation</key>
    <string>Portrait</string>
    <key>UILaunchImageSize</key>
    <string>{768, 1024}</string>
  </dict>
  <dict>
    <key>UILaunchImageMinimumOSVersion</key>
```

```

<string>7.0</string>
<key>UILaunchImageName</key>
<string>LaunchImage-iOS7</string>
<key>UILaunchImageOrientation</key>
<string>Portrait</string>
<key>UILaunchImageSize</key>
<string>{320, 568}</string>
</dict>
<dict>
<key>UILaunchImageMinimumOSVersion</key>
<string>7.0</string>
<key>UILaunchImageName</key>
<string>LaunchImage-iOS7</string>
<key>UILaunchImageOrientation</key>
<string>Portrait</string>
<key>UILaunchImageSize</key>
<string>{320, 480}</string>
</dict>
</array>

```

در ادامه وارد فایل pro. شده و دستور زیر را اعمال کنید:

```

app_launch_images.files = $$files($$PWD/ios/icons/LaunchImage*.png)
QMAKE_BUNDLE_DATA += app_launch_images

```

## انتشار برنامه در Apple App Store

بر اساس مواردی که به آن‌ها اشاره شد برنامه شما آماده انتشار بر روی فروشگاه اپل خواهد بود که تحت آدرس فروشگاه اپل در دسترس است. توسط دستورات فوق برنامه شما طبق استاندارد اپل بر روی Apple Store قابل پذیرش خواهد بود، برای ساخت فایل ipa جهت ارسال بر روی آپ استور و استفاده از طریق iTunes پوشاهای با نام Payload ایجاد کنید و فایل MyApp.app را در آن قرار دهید. در نهایت پوشه Payload را فشرده کنید (Zip) سپس فایل زیپ شده را تغییر نام و تغییر پسوند دهید برای مثال MyApp.ipa سپس تایید کنید تا فایل آماده نصب بر روی دستگاهها و ارسال بر روی Apple App Store آماده شود. البته دقت کنید که اگر حساب کاربری شما شارژ نباشد و تایید نشده باشد فایل ipa بر روی دستگاه اجرا نخواهد شد.

**نکته:** حداقل نسخه قابل قبول برای انتشار بر روی فروشگاه اپل ۷.۰ است.

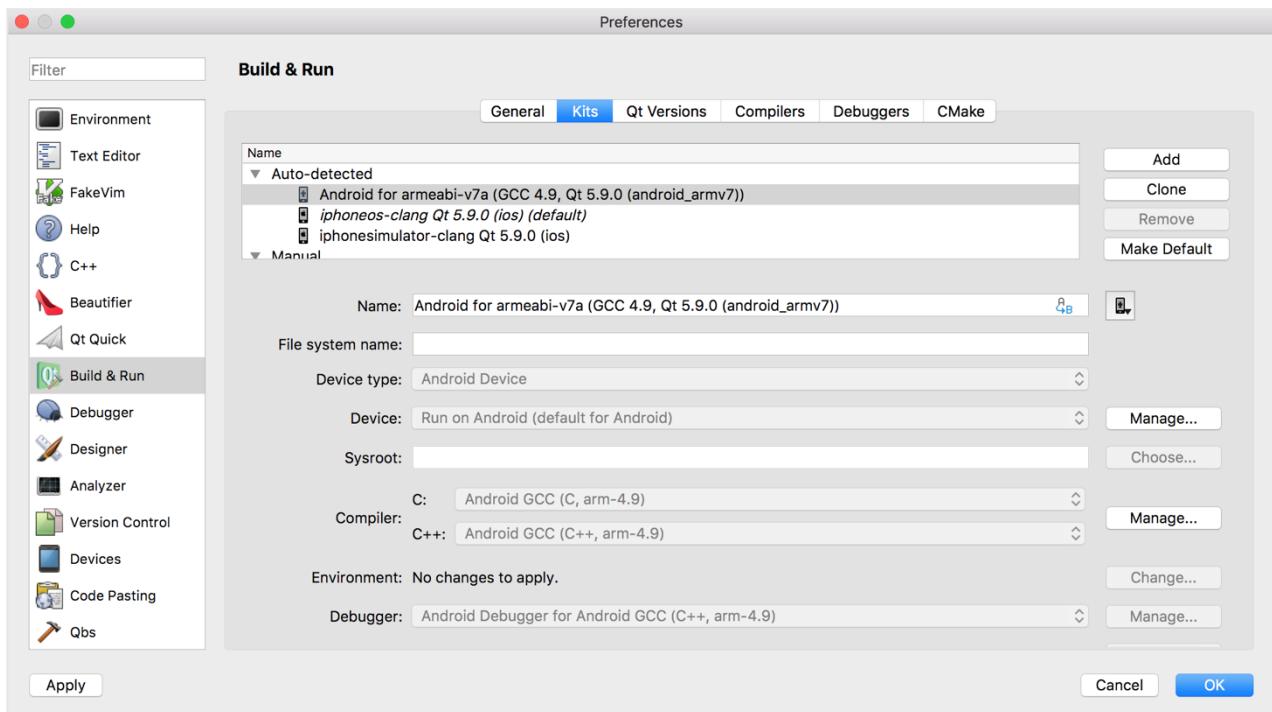
## پیکربندی و انتشار برنامه در پلتفرم گوشی‌ها و تبلت‌های هوشمند تحت اندروید (Android)

همانطور که در رابطه با توسعه برنامه بر روی iOS توضیح داده شد اندروید نیز پیکربندی خاصی را در بخشی از موارد برای خود دارد که شما باید جهت خروجی گرفتن برای اندروید مواردی را از قبل در نظر بگیرید. قبل از هر چیز باید بدانیم که ساختار بسته برنامه اندرویدی در قالب فشرده Zip است که تحت نام APK معرفی می‌شود. البته در صورتی که نیاز باشد از منابع دیگری مانند کدهای Java استفاده شود می‌توانید آن‌ها را در مسیر زیر در اختیار برنامه قرار دهید.

مهمنترین قسمت مربوط به پیکربندی برنامه‌های اندروید فایل **AndroidManifest.xml** است که توسط دستگاه‌های مجهز به این سیستم‌عامل استفاده می‌شود. این فایل اجازه می‌دهد تا ویژگیها، دسترسی و بسیاری از خصیت‌های برنامه را بر روی سیستم‌عامل مشخص کرد. جهت کسب اطلاعات بیشتر در این مورد گوگل مستندات کاملی را در اختیار شما قرار خواهد.

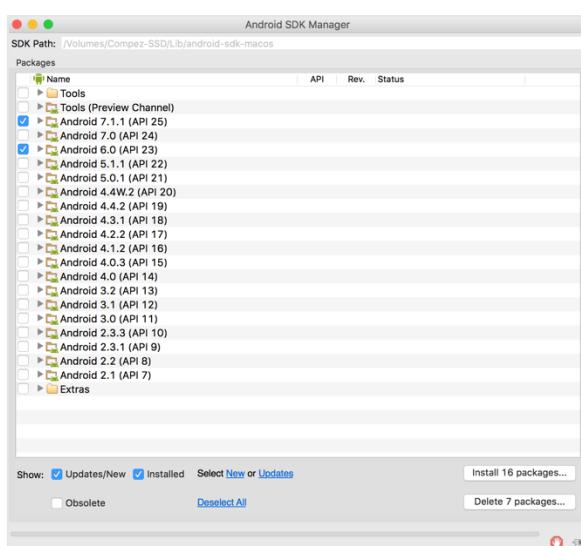
## اتصال به دستگاه تحت اندروید جهت ساخت، آزمایش و انتشار برنامه

در صورتی که نسخه مرتبط با اندروید کیوت را نصب کرده باشید که در فصل اول کتاب به نسخه‌های مرتبط آن اشاره شده است. در زبانه Kits به صورت زیر نوع کامپایلر برای اندروید شناسایی خواهد شد.



توجه داشته باشید که پیش نیازات برای دسترسی به این مورد پکیج‌های مرتبط با JDK نسخه ۶ به بالا و همچنین ابزارهایی برای ساخت برنامه برای اندروید مانند Apache Ant نسخه ۱.۸.۰ به بالا و NDK و SDK نیز برای فراهم آوردن ابزارهای ساخت و توسعه در پلتفرم اندروید نیاز خواهد بود که باید آن‌ها را نصب و به روز رسانی نمایید.

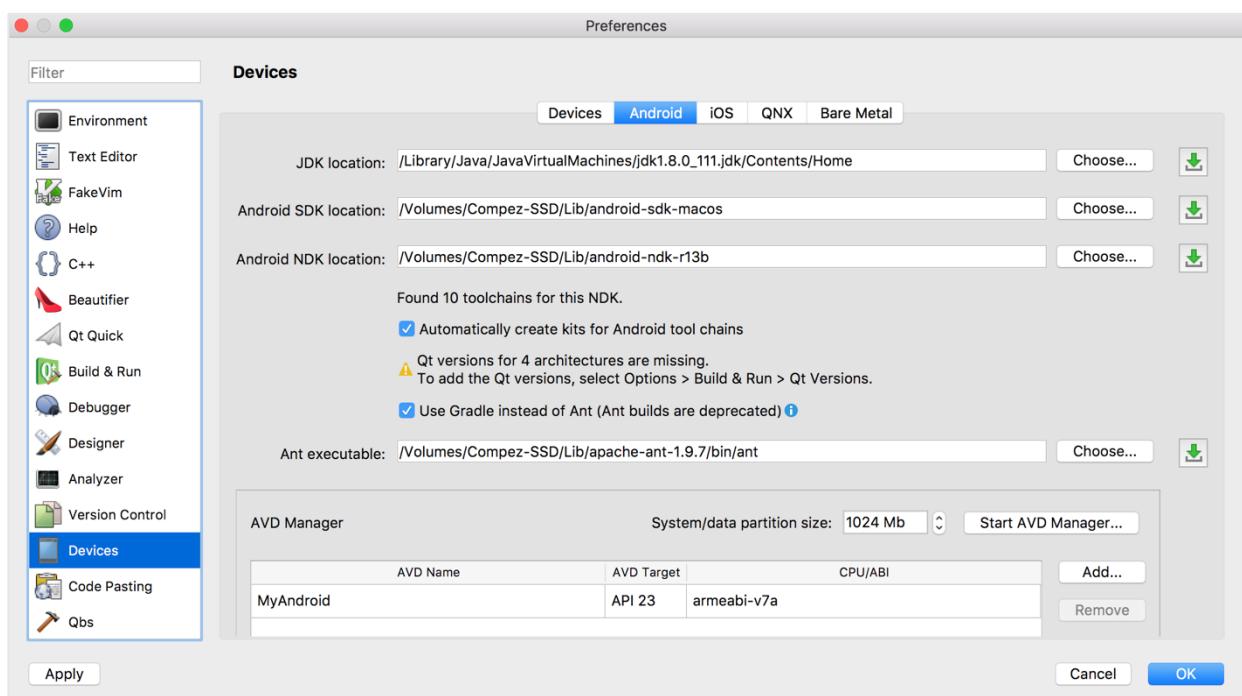
بسته توسعه و رابطه‌های برنامه‌نویسی اندروید که در پکیج SDK قرار دارد در مسیر `/android-sdk/tools/android ...` را برای اجرای Android فراهم می‌آورد. در صورتی که فایل اجرای android را در مسیر فوق اجرا کنید محیط مدیریت SDK ظاهر می‌شود که به صورت زیر خواهد بود که جهت دریافت مخازن و به روز رسانی پکیج‌ها و API‌های مرتبط با سیستم‌عامل اندروید حتماً باید با استفاده از VPN اقدام به دریافت آن کنید.



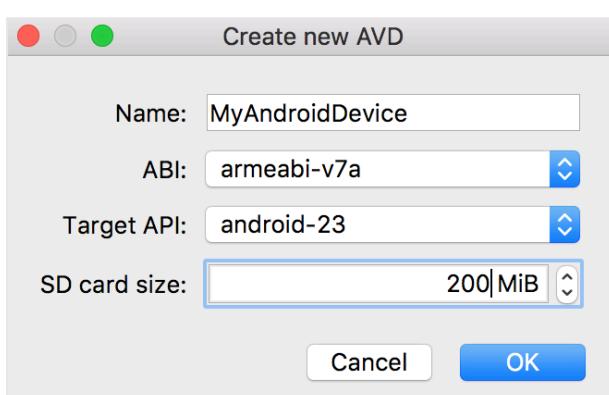
این بخش برای توسعه‌دهندگان زبان جاوا هم همینگونه خواهد بود. زیرا قبل از آغاز برنامه‌نویسی اندروید نیاز به پکیج‌ها و رابطه‌ای برنامه‌نویسی مورد نظر اندروید هستیم. بنابراین با انتخاب برخی از پکیج‌های مورد نظر بر اساس نیاز اقدام به روز رسانی آن خواهیم کرد که ما نسخه ۶ و ۷ با شماره رابطه‌ای ۲۳ و ۲۵ انتخاب کرده‌ایم.

البته توصیه می‌شود حتیماً پکیج نسخه ۱۸ را که شامل رابطه‌ای ۴.۳.۱ است نصب کنید. دقت کنید که این مراحل در ویندوز، لینوکس و مک یکسان است تنها کافی است شما فایل اجرایی android را از داخل پکیج انتخاب و اجرا کنید.

توجه کنید که بعد از دریافت فایل‌های پیش نیاز که به آن‌ها اشاره شد باید به قسمت Devices و زبانه Preferences رفته و به قسمت Android NDK و Apache Ant را شناسایی کنیم. این کار به صورت دستی صورت خواهد گرفت. مراجعه کنید. در این بخش قرار است مسیرهای JDK، SDK، Apache Ant و NDK را شناسایی کنیم. این کار به صورت دستی صورت خواهد گرفت.



دقت کنید که مسیرهای فوق صحیح باشد در غیر اینصورت پیغام خطاب برای هریک صادر خواهد شد که شما باید آن را رفع نمایید. بر روی گزینه Add در بخش AVD Manager کلیک کنید تا به صورت زیر یک دستگاه مجازی اندرویدی ایجاد شود.

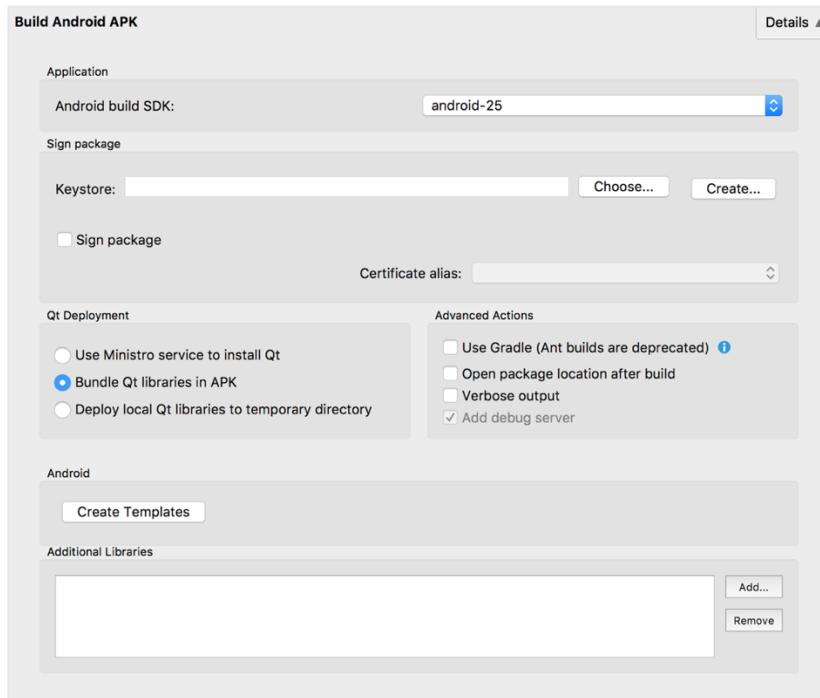


در این بخش برای اینکه برنامه خود را بر پایه دستگاه‌های واقعی بررسی و مورد آزمایش قرار دهید بهتر است ABI را بر روی armeabi-v7a تنظیم کنید. گزینه Target API نشان می‌دهد که برنامه شما بر اساس کدام نسخه از API موجود در اندروید پشتیبانی خواهد شد. در نهایت گزینه Apply

و OK را بزنید و برای ساخت پروژه و ارسال خروجی به قسمت Projects رفته و گزینه مربوط به اندروید را برای پیکربندی ساخت و توسعه برای استقرار انتخاب کنید.

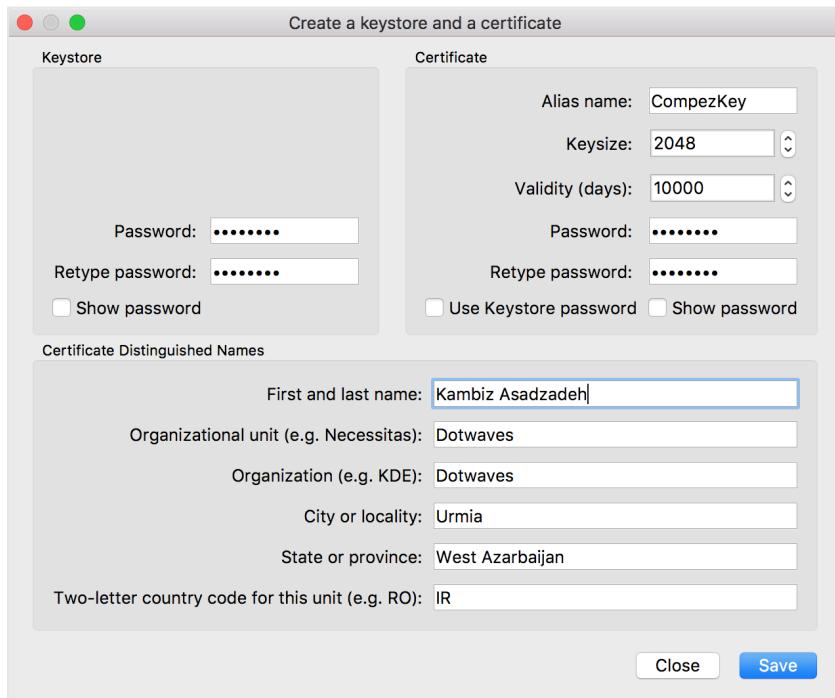


در قسمت Projects به زبانه Android build SDK رفته و Build Android APK را برابر یکی از نسخه‌های موجود API قرار دهید که طبق تصویر ما از نسخه ۲۵ مختص اندروید ۶ استفاده کرده‌ایم.



در قسمت Projects به زبانه Android build SDK رفته و Build Android APK را برابر یکی از نسخه‌های موجود API قرار دهید که طبق تصویر ما از نسخه ۲۵ مختص اندروید ۶ استفاده کرده‌ایم. در زبانه Sign Package برای انتشار برنامه‌های خود، شما باید آن را توسط یک جفت کلید عمومی و خصوصی از یک گواهی و یک کلید متناظر که با یک نام مستعار شناخته می‌شود ایجاد کنید. این جفت کلید جهت بررسی این که نسخه‌های آینده از نرم‌افزار شما مورد تایید قرار بگیرند ایجاد می‌شود.

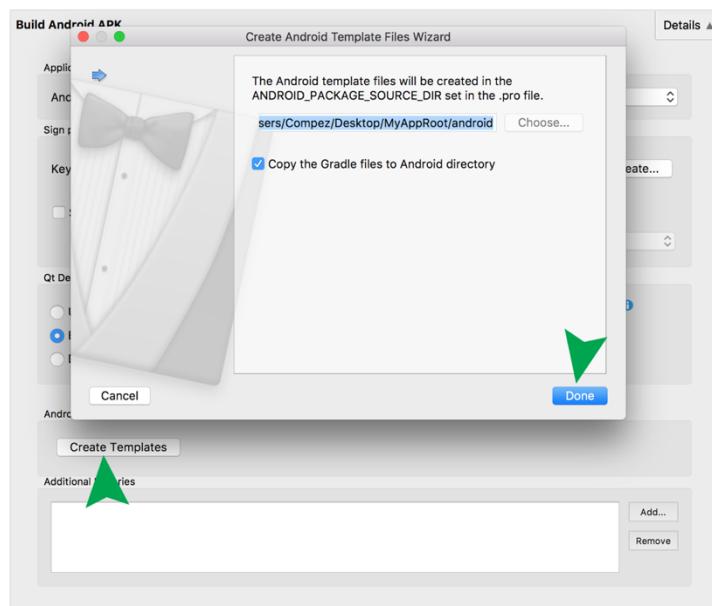
نکته: سعی کنید کلید خود را در یک جای امن نگهداری کنید چرا که در صورت از دست دادن آن امکان به روز رسانی نرم‌افزار شما غیرممکن خواهد بود. این بخشی از امنیت توسعه در برنامه‌های اندرویدی به شمار می‌رود.



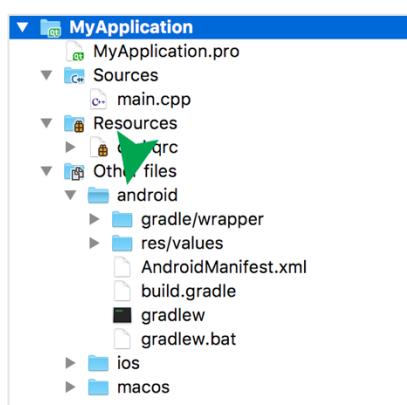
در صورت نیاز طبق تصویر فوق می‌توانید اطلاعات خود را همراه با مقادیر امنیتی ایجاد کنید. در صورتی که کلید فوق را ساخته باشید می‌توانید برای هربار به روز رسانی برنامه آن را انتخاب و استفاده کنید.

- در زبانه **Use Minstro service to install Qt** گزینه **Qt Deployment** در صورتی انتخاب می‌شود که نیاز نباشد کتابخانه‌های کیوت همراه بسته APK مستقر شود. در این صورت برنامه به صورت خودکار و آنلاین قبل از اجرا بسته‌های مورد نیاز را دریافت و نصب خواهد کرد. این امر باعث می‌شود حجم فایل APK بسیار پایین باشد اما در عوض کاربرانی که آن را دانلود می‌کنند حتماً یک بار کتابخانه‌های مورد نیاز را بر روی اندروید نصب خواهند کرد.
- با انتخاب گزینه **Boundle Qt libraries in APK** تمامی کتابخانه‌های مورد نیاز Qt همراه فایل APK منتشر خواهند شد که در این صورت کاربر نیازی به نصب کتابخانه نخواهد داشت اما حجم برنامه کمی بیشتر از نوع اول خواهد بود.
- با انتخاب گزینه **Deploy local Qt libraries to temporary directory** تمامی کتابخانه‌های مورد نیاز Qt در مسیر موقت /data/local/tmp/qt نصب خواهند شد.
- در زبانه **Advanced Action** گزینه **Use Gradle** ابزاری است که برای ساخت برنامه مورد استفاده قرار می‌گیرد. اما این گزینه پیشنهاد نمی‌شود چرا که در نسخه‌های بعدی اندروید منسخ خواهد شد.
- با انتخاب گزینه **Open package location after build** در زمان به اتمام رسیدن مراحل ساخت بعد از کامپایل محل (دایرکتوری) ایجاد فایل apk نمایان خواهد شد.
- در صورت انتخاب گزینه **Verbose Output** در زمان کامپایل رخدادهایی که توسط ابزار **andddroiddeployqt** صورت می‌گیرد را در کنسول Qt Creator مشاهده خواهید کرد.
- گزینه **Add Debug Server** سرور مربوط به اشکال زدایی را به برنامه اضافه می‌کند که به طور پیشفرض در حالت انتخاب است.

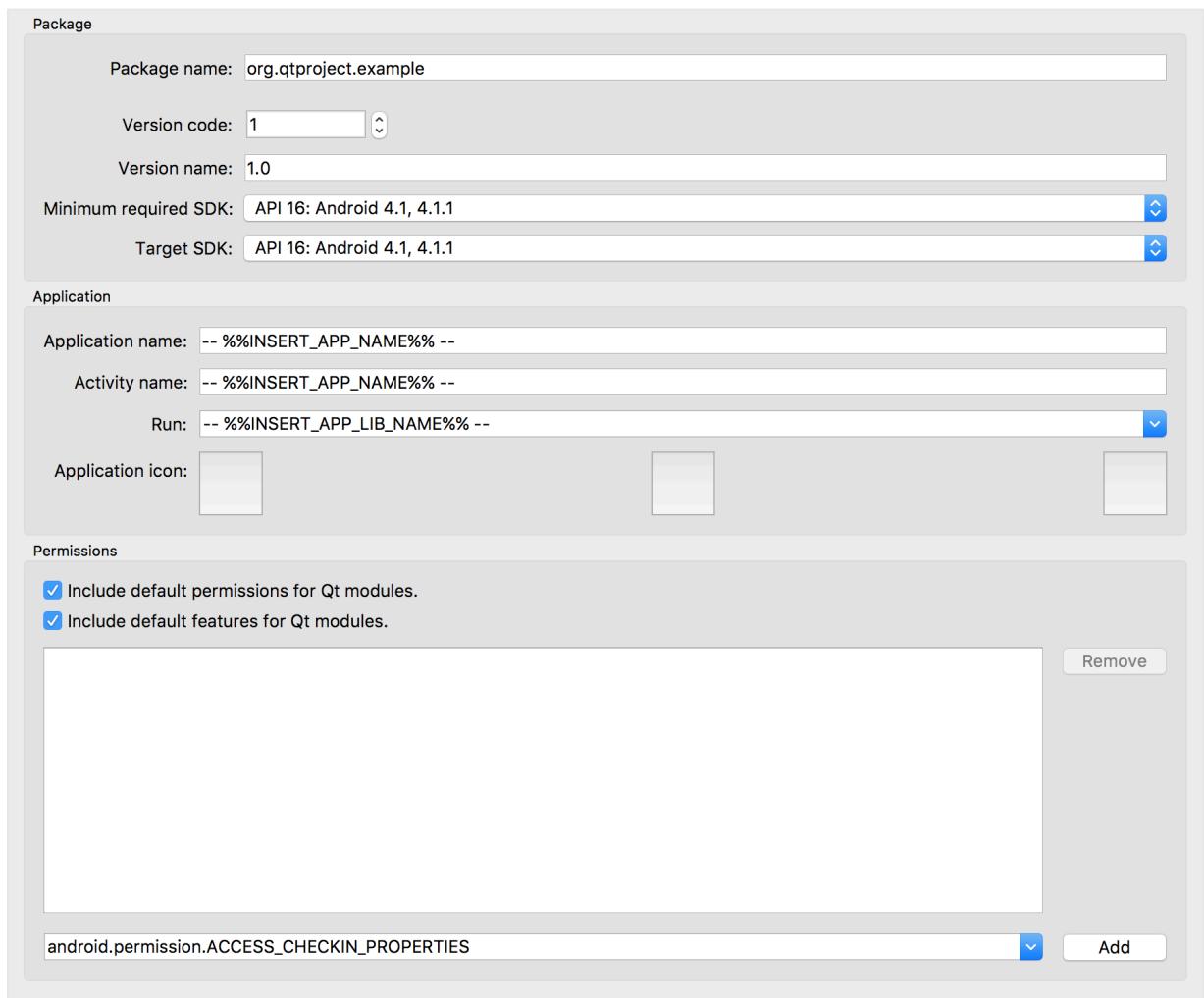
زبانه **Android** و گزینه **Create Template** یکی از مهمترین بخش‌های توسعه برنامه اندروید را رقم می‌زند. با انتخاب این گزینه امکان ساخت، کپی و مدیریت فایل **AndroidManifest.xml** فراهم می‌شود.



کافی است آن را کپی و اعمال نمایید تا به گزینه‌های اندروید دسترسی داشته باشد. در صورتی که اعمال شود در پوشه android فایل‌هایی ایجاد خواهد شد.



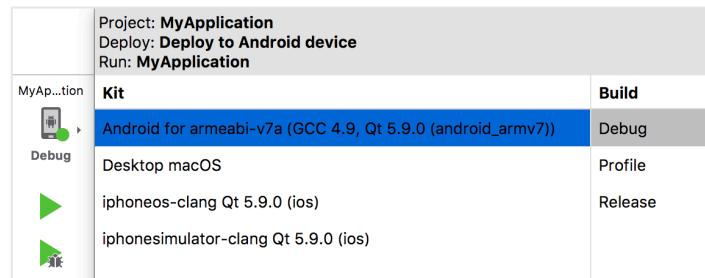
که با کلیک بر روی **AndroidManifest.xml** محیط مدیریت آن نمایان می‌شود. محیط مدیریتی فایل مانیفست شامل بخش Application, Package و Permission است که به ترتیب برای مشخص سازی اطلاعات اپلیکیشن و متغیرهای مربوط به نام، اکتیویتی، آیکون است. در نهایت بخش مجوزها یکی از مهمترین قسمت‌هایی است که در توسعه برنامه‌های اندروید کاربرد خواهد داشت. این محیط برخلاف محیط توسعه iOS کمی ساده‌تر است.



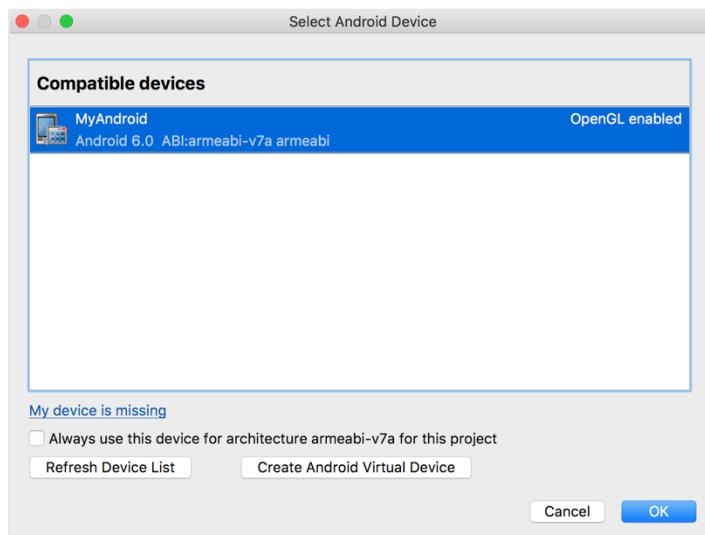
محیط مدیریتی فایل مانیفیست شامل بخش Package و Permission است که به ترتیب برای مشخص سازی اطلاعات اپلیکیشن و متغیرهای مربوط به نام، اکتیویتی، آیکون است. در نهایت بخش مجوزها یکی از مهمترین قسمت‌هایی است که در توسعه برنامه‌های اندروید کاربرد خواهد داشت. نام پکیج بر اساس استاندارد تعریف می‌شود. برای مثال پروژه ما تحت نام com.genyleap.android ایجاد می‌شود. این بر اساس سلیقه توسعه دهنده معرفی می‌شود که در مدیریت سرویس نیز از آن استفاده خواهد شد. کد نسخه و مقدار نسخه برنامه و همچنین حداقل و حداکثر پشتیبانی از API مورد نظر در این بخش قابل تنظیم است.

گزینه‌های include default features for Qt modules و include default permissions for Qt modules بهتر است به حالت انتخاب باشند چرا که تمامی ویژگی‌ها بر روی اندروید با فعال‌سازی این دو گزینه امکان‌پذیر خواهند بود. همچنین در برنامه‌نویسی اندروید ایجاد دسترسی برای برنامه بسیار مهم است برای مثال دسترسی برای اینترنت توسط برنامه باید در این قسمت مشخص شود. البته لازم به ذکر است که جهت مدیریت حرفه‌ای تر بهتر است بر روی فایل مانیفیست راست کلیک کرده و گزینه Open With->Plain Text Editor را انتخاب کنیم.

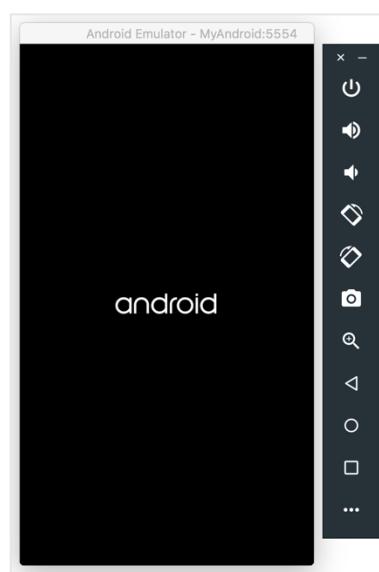
این محیط برخلاف محیط توسعه iOS کمی ساده تر است. در صورتی که مراحل زیر را انجام داده‌اید گزینه qmake را برای پیش‌سازی برنامه اندروید اجرا کنید. توجه داشته باشید که نوع کامپایلر شما بر روی نسخه اندروید تنظیم شده باشد.



بعد از کامپایل برنامه را اجرا کنید تا مرحله زیر را مشاهده نمایید در این بخش نسخه مجازی اندروید که ساخته‌ایم را مشاهده می‌کنیم، به راحتی آن را انتخاب و سپس بر روی OK کلیک کنید تا شبیه ساز پس از مدت زمانی کوتاه اجرا و برنامه شما را نمایش دهد.



نکته قابل توجه: کامپایل برنامه‌های اندروید کمی زمانبر می‌باشند که به خاطر مجازی بودن سیستم اندروید بر روی موتور جاوا است. بنابراین کمی حوصله به خرج دهید و سپس برنامه خود را بررسی نمایید.



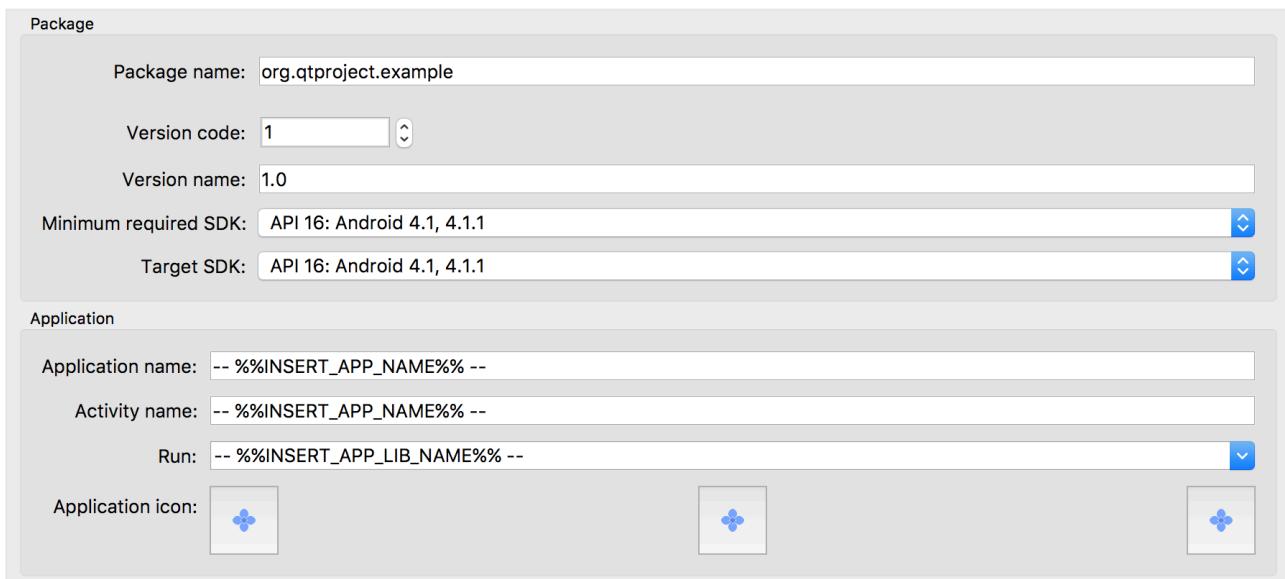
اگر می‌خواهید بر روی دستگاه واقعی برنامه خود را اشکال زدایی کنید بر روی گوشی خود امکان **Developer Mode** را فعال کنید. دقت کنید که **USB Debugging** فعال باشد. وارد قسمت **About** گوشی خود شوید و از قسمت **Status** آدرس آی پی خود را یادداشت کنید. برای مثال با استفاده از دستور **adb** امکان افزودن شناسه دستگاه به لیست دستگاه‌های شناسایی شده فراهم می‌شود.

**adb connect 192.168.x.x**

در نهایت پیغام connected to 192.168.x:5555 را دریافت خواهید کرد. البته دقت کنید که آدرس مربوطه در هر دستگاه متفاوت خواهد بود. اینبار برنامه را اجرا کنید تا نام دستگاه شما در بین لیست انتخاب شود.

## آیکون برنامه در Android

همانطور که در نسخه iOS ایجاد آیکون برای برنامه را دیدیم در این بخش نیز برای برنامه اندروید می‌توانیم این روند را تکرار کنیم. اما به دلیل آسودگی مدیریت تحت فایل مانیفیست در اندروید این کار در خود Qt Creator امکانپذیر است.



در قسمت Application سه مقدار Application icon را از مسیر مورد نظر مقدار دهی نمایید و آیکون‌های خود را برای آن معرفی کنید. بعد از کامپایل و نصب بر روی گوشی برنامه شما دارای آیکون مختص خود خواهد بود. دقت کنید که کیفیت فایل آیکون بسیار مهم است.

## افزودن کتابخانه‌های نوع سوم بر پایه جاوا در کیوت

بسیاری از رابطه‌های برنامه‌نویسی تحت جاوا برای اندروید از طرف گوگل وجود دارند که بسیار کاربردی هستند. بنابراین برای دسترسی به آن‌ها امکان آن توسط متغیر ANDROID\_PACKAGE\_SOURCE\_DIR با تعریف در فایل pro. امکان‌پذیر است. البته اگر شما از Qt Creator استفاده کنید تنها با ساخت فایل AndroidManifest.xml کد مربوط به آن در فایل pro. ایجاد خواهد شد.

**ANDROID\_PACKAGE\_SOURCE\_DIR = \$\$PWD/ANDROID**

برای مثال برخی از این کتابخانه‌ها همراه با SDK Google Play Service نصب می‌شوند که در زیر شاخه پروژه که توسط متغیر \$ANDROID\_SDK\_ROOT/extras/google/google\_play\_services\_libproject/google-play-services\_lib مشخص می‌شود. در نهایت همان پوشه همراه با فایل‌ها را در زیر پروژه تحت کیوت کپی خواهیم کرد.

**\$PROJECT\_ROOT/android/google-play-services\_lib**

## نکات قابل توجه در اندروید

- تمامی برنامه‌های کیوت ۵ در نسخه‌های اندروید ۴.۱ به بالا با رابط برنامه‌نویسی حداقل ۱۶ قابل اجرا هستند و البته تمامی ماژول‌ها به جز ماژول **QtWebEngine** قابل پشتیبانی هستند.

- پشتیبانی از سیستم فایل مجازی در اندروید بر پایه مکانیزم assets است. تمامی فایلهایی که در داخل پوشه assets کپی می‌شوند بخشی از بسته‌های موجود در پروژه خواهند بود. برای مثال ممکن است لوگوی برنامه در زیر assets:/images/logo.png قرار گیرد. البته پشتیبانی از مکانیزم فایل مجازی تحت کیوت نیز میسر است که تخت فایل qrc فراهم می‌شود.
- نسخه‌های قابل پشتیبانی در اندروید تحت کیوت معماری‌های x86 و ARMv7 می‌باشند. بنابراین اگر برنامه شما تحت کامپایلر x86 کامپایل و بر روی معماری ARM اجرا شود بدون شک خطا خواهد داد.

## دریافت فایل apk در قالب یک بسته کامل

بعد از کامپایل برنامه به پوشه android-build وارد شده و داخل پوشه bin شوید. در این قسمت فایل با نام **QtApp-debug.apk** را انتخاب کنید و بر روی گوشی خود نصب نمایید.

## انتشار برنامه بر روی Google Play

بعد از اینکه تصمیم نهایی برای انتشار برنامه بر روی فروشگاه گوگل گرفتید کافی است مراحل زیر را انجام دهید:

۱. توسط کیوت کریتور پروژه را باز کرده حالت کامپایل برنامه را بر روی Release قرار دهید، سپس کامپایل کرده و از آن خروجی apk دریافت کنید.
۲. دقیق کنید که حتماً فایل AndroidManifest.xml را ایجاد کرده و سپس مقادیر مرتبط با آن را به طور صحیح و کامل معین کنید. فراموش نکنید زمانی که حالت کامپایل را تغییر می‌دهید حتماً نیاز خواهد بود تا این فایل را مجدداً پیکربندی کنید.
۳. حداقل SDK لازم را بر روی نسخه ۹.۰ قرار دهید.
۴. در صورتی که از استایل (پوسته) بومی اندروید استفاده می‌کنید حتماً باید API شما نسخه ۱۱ به بالا باشد.
۵. مقادیر Application Name و Application Icon را حتماً مقدار دهی کنید.
۶. تمامی دسترسی‌هایی که برنامه شما به آن نیاز دارد را مشخص کنید.
۷. گزینه‌های مرتبط با ویژگی‌های کیوت مثل NFC و غیره را فعال کنید.
۸. در این مرحله پیشنهاد می‌شود تا از ویژگی keystore استفاده شود، آن را تنظیم کرده و گزینه Open package location after build را انتخاب کنید.
۹. در نهایت فایل apk خروجی را دریافت کرده و به داشبورد (کنسول) گوگل در آدرس (<https://play.google.com/apps/publish>) وارد شوید.
۱۰. در صورتی که اکانت گوگل شما معتبر باشد و قوانین انتشاری آن را رعایت کنید برنامه شما قابل استقرار بر روی فروشگاه جهانی گوگل خواهد بود. این روند را برای وبسایت‌هایی مانند کافه بازار انجام دهید.

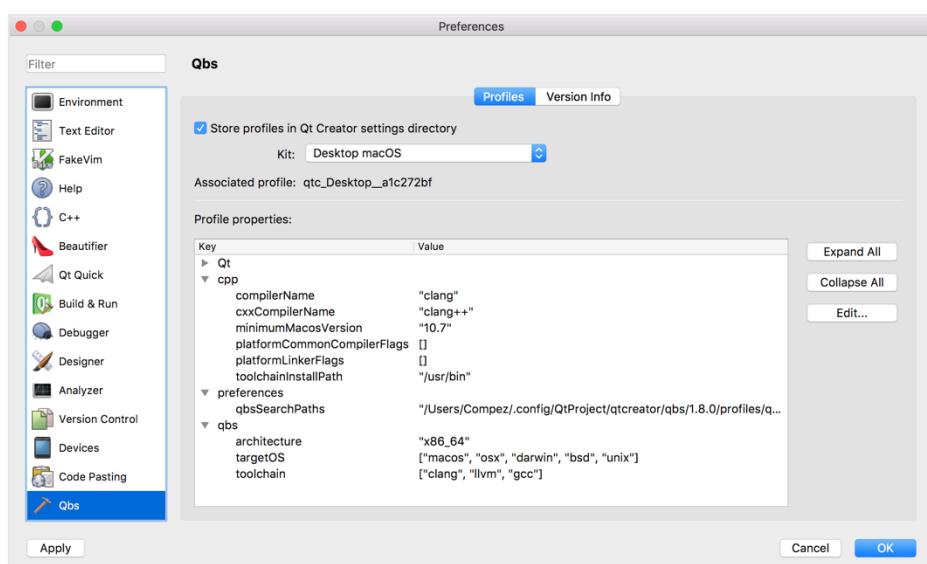
## معرفی ابزار کیوبس (QBS) و پیکربندی آن

کیوبس (Qbs) با آوای (“Cubes”) همانند qmake یک ابزار بسیار ساده و قدرتمند برای ساخت پروژه در پلتفرم‌های مختلف است. این ابزار قابلیت استفاده در هر نوع پروژه‌ای را دارد بگونه‌ای که مهم نیست زبان برنامه‌نویسی، ابزارها و حتی کتابخانه‌های مورد استفاده چه چیزی باشند. کیوبس یک ابزار همه کاره است که یک نمودار ساخت از پروژه‌های سطح بالا مانند (cmake و qmake) فراهم می‌کند که علاوه بر آن وظیفه اجرای دستورات نمودار ساخته شده در سطح پایین را مانند (make) فراهم می‌کند. در این کتاب ما به پیش معرفی آن می‌پردازیم چرا که قرار است در نسخه ۶ کیوت جایگزین نوع pro. باشد.

کیوبس برنامه‌ها را بر پایه اطلاعاتی می‌سازد که در قالب یک فایل QML فراهم می‌شوند. هر فایل پروژه مشخص می‌کند که پروژه می‌تواند شامل چندین محصول باشد و شما نوع پروژه می‌توانید مشخص کنید که یکی از انواع : نرم افزار، کتابخانه و غیره باشد. مشخصات سیستم مورد نیاز جهت استفاده از ابزار QBS حداقل نسخه ۵.۶.۰ به بعد خواهد بود.

### معرفی ساختار QBS در قالب QML

کیوبس برای فایل پروژه خود از پسوند (\*.qbs) استفاده می‌کند تا بتواند محتوای موجود در یک پروژه را معرفی کند. به طور کلی یک پروژه شامل چندین محصول (پروژه) است که در ادامه به آن‌ها اشاره شده است. یک محصول هدفی از روند ساخت است، به طور معمول یک برنامه کاربردی، کتابخانه و یا شاید یک چیز دیگر می‌تواند هدف آن باشد. جهت بررسی و مدیریت Qbs وارد بخش Settings یا Preferences شوید و زبانه Qbs را انتخاب کنید تا در این بخش به نسخه و پیکربندی مقادیر مربوط به Qbs دسترسی داشته باشید.



توجه داشته باشید که فایل‌های منابع Qbs به صورت UTF-8 کدگذاری شده‌اند.

همانطور که ابتدا به آن اشاره شده است ساختار Qbs بر پایه گویش و سبک QML است که توسط آن یک پروژه ساده برای اجراء پیغام “سلام، دنیا!” به صورت زیر خواهد بود:

```
import qbs 1.0

Application {
    name: "helloworld"
```

```

    files: "main.cpp"
    Depends { name: "cpp" }
}

```

برخلاف pro.qbs. امکان تعریف مازول‌ها، سورس فایل‌ها و دستورات دیگر به راحتی امکان‌پذیر است. به عنوان مثال فرض بر اینکه ما نام، نوع، فایل اصلی پروژه و وابستگی به نوع آن را در زیر مشخص کرده‌ایم.

```

Product {
    name: "helloworld"
    type: "application"
    files: "main.cpp"
    Depends { name: "cpp" }
}

```

بخشی که باید بسیار به آن توجه کرد نوع (type) است که نوع پروژه را برای خروجی مشخص می‌کند. مثلاً لازم است پروژه به صورت فایل اجرایی ایجاد شود یا به صورت کتابخانه در قالب استاتیک یا پویا یا پروژه خاص C++ که همه آن‌ها را می‌توان به راحتی از قبل مشخص کرد. مقادیر قابل استفاده در نوع type به صورت زیر هستند:

- Application
- CppApplication
- DynamicLibrary
- StaticLibrary

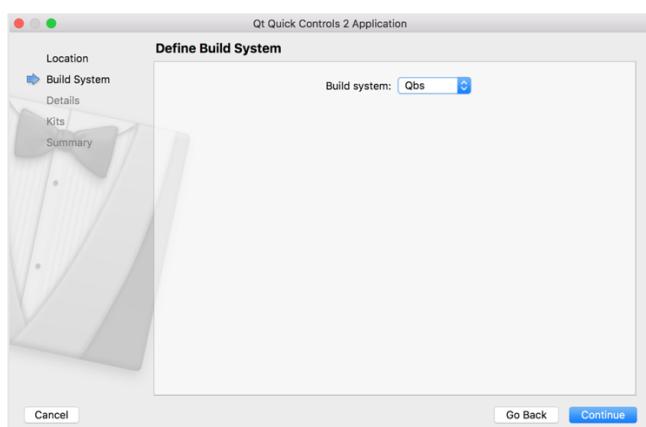
به عنوان مثال نوع پروژه C++ به صورت زیر تعریف می‌شود:

```

CppApplication {
    name: "helloworld"
    files: "main.cpp"
}

```

جهت استفاده از QBS در پروژه خود کافی است در مرحله ایجاد یک پروژه در بخش Build System نوع آن را روی qbs تنظیم کنید و سپس پروژه خود را ایجاد کنید.



اگر مثال جامع‌ای در رابطه با محتوای موجود در یک پروژه تحت QBS داشته باشیم به صورت زیر خواهد بود:

```

import qbs 1.0

Project {

```

```

minimumQbsVersion: "1.7.1"

CppApplication {
    Depends {
        name: "Qt.core"
    }
    Depends {
        name: "Qt.quick"
    }

    property pathList qmlImportPaths: []

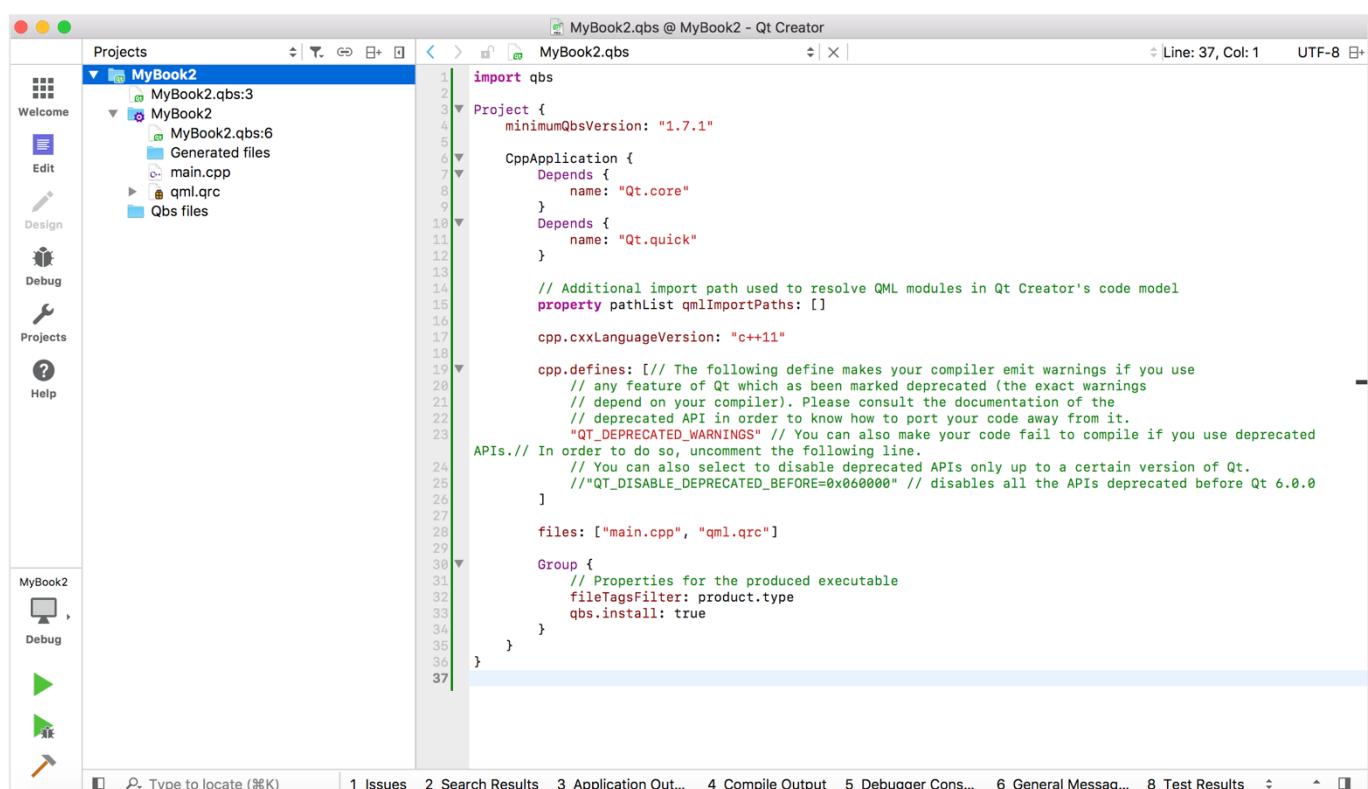
    .cppcxxLanguageVersion: "c++14"

    .cppdefines: [
        "QT_DEPRECATED_WARNINGS"
    ]

    files: ["main.cpp", "qml.qrc"]

    Group {
        fileTagsFilter: product.type
        qbs.install: true
    }
}
}

```



دقت کنید که در این نوع پروژه خبری از فایل pro. که مهمترین فایل پروژه در کیوت است. استفاده از qbs یک روش بسیار مهم و کارآمد است که در نسخه‌های جدید کیوت قرار است استفاده شود. این ابزار اجازه می‌دهد تا با سرعت و دقت بسیار بیشتری تجربه خوبی را جهت مدیریت پروژه خود و پیکربندی آن برای توزیع کسب نمایید.

توجه داشته باشید که قبل از هرچیز تمامی مقادیر و تعاریف می‌توانند زیر مجموعه Project را می‌توان به صورت انحصاری تعریف نموده و سپس توسط Depends مازول‌های مورد نظر را برای آن فراخوانی کنیم که همانند `=+ QT` در ساختار pro. عمل می‌کند برای مثال مقدار QtQuick مازول را فراخوانی می‌کند.

خاصیت pathList لیستی از مقادیر مسیرها را مشخص می‌کند، اگر دو فایل داشته باشیم به صورت زیر pathList مقدار دهی می‌شود:

```
["file1.txt", "./file2.txt"]
```

خاصیت cppcxxLanguageVersion. در QBS مختص مشخص سازی نسخه زبان برنامه‌نویسی C++ است که مشخص می‌کند پروژه تحت چه نسخه‌ای از زبان برنامه‌نویسی سی‌پلاس‌پلاس ساخته خواهد شد که این مورد نیز در pro. توسط متغیر CONFIG قابل مقدار دهی بود. در کنار آن است در پروژه استفاده شود. همچنین نوع Group موارد مورد نیاز را در قالب گروه‌های مختلف فراهم می‌کند. برای مثال قصد داریم فایل‌هایی را ایجاد کنیم که هریک از آن‌ها برای اهداف خاص و یا حتی پلتفرم‌های خاص و اشتراکی قابل استفاده خواهند بود. در کیوبس امکان مدیریت گروهی آن‌ها بسیار شکیل‌تر و مدیریت شده تر است.

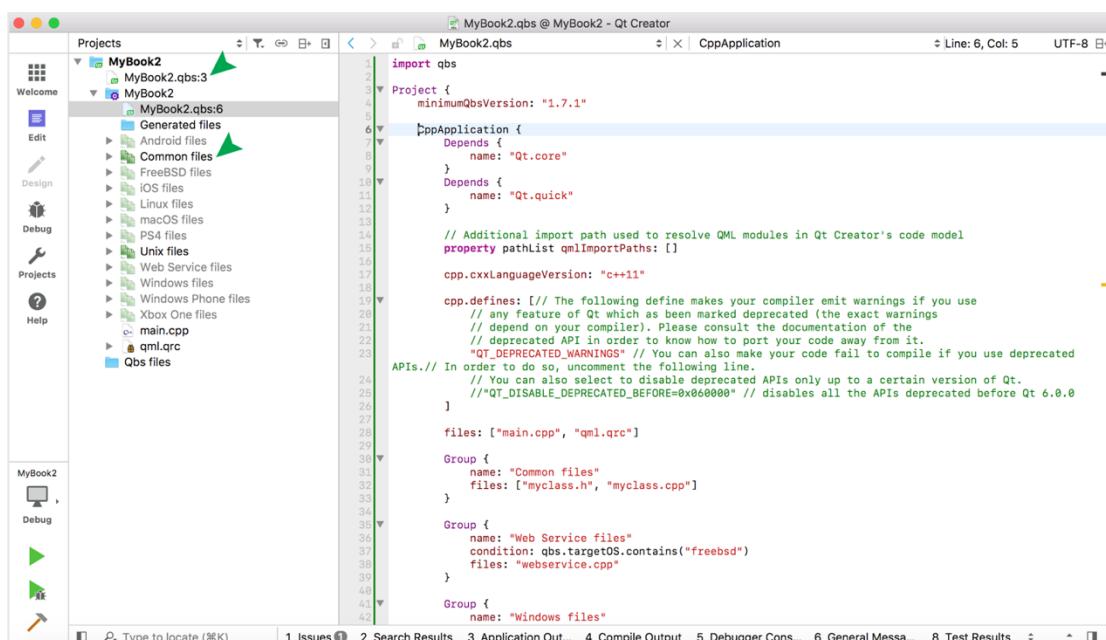
```
Group {
    name: "Common files"
    files: ["myclass.h", "myclass.cpp"]
}
Group {
    name: "Web Service files"
    condition: qbs.targetOS.contains("freebsd")
    files: "webservice.cpp"
}
Group {
    name: "Windows files"
    condition: qbs.targetOS.contains("windows")
    files: "mywindowsclass.cpp"
}
Group {
    name: "Windows Phone files"
    condition: qbs.targetOS.contains("winrt")
    files: "winrt.cpp"
}
Group {
    name: "Xbox One files"
    condition: qbs.targetOS.contains("xbox")
    files: "xbox.cpp"
}
Group {
    name: "Unix files"
    condition: qbs.targetOS.contains("unix")
    files: "myunixclass.cpp"
}
Group {
    name: "macOS files"
    condition: qbs.targetOS.contains("linux")
    files: "macosclass.cpp"
}
Group {
    name: "iOS files"
    condition: qbs.targetOS.contains("ios")
    files: "ios.cpp"
}
Group {
```

```

name: "Linux files"
condition: qbs.targetOS.contains("linux")
files: "mylinuxclass.cpp"
}
Group {
    name: "FreeBSD files"
    condition: qbs.targetOS.contains("freebsd")
    files: "myfreebsd.cpp"
}
Group {
    name: "PS4 files"
    condition: qbs.targetOS.contains("freebsd")
    files: "myps4.cpp"
}
Group {
    name: "Android files"
    condition: qbs.targetOS.contains("linux")
    files: "android.cpp"
}
}

```

با توجه به ساختار و مقادیر تعریف شده name و condition در **Group** مدیریت هریک از آن‌ها نتیجه‌های به صورت زیر را خواهد داشت که برخلاف نوع pro، بسیار کاربر پسند و کاربردی از لحاظ مدیریت و دسترسی به اهداف مورد نظر در توسعه است:



در یک نگاه سریع اعدادی که در مقابل پسوند qbs قابل مشاهده هستند نشانگر این است که چه تعداد آیتم در گروه‌ها در نظر گرفته شده‌اند. متغیر condition دستوران شرطی را نگه‌دارد که در این بخش با استفاده از تابع از پیش تعریف شده در qbs مقادیر در آن ارسال و بر اساس نوع پلترم دستورالعمل‌ها فراخوانی می‌شوند.

همچنین متغیرهای name جهت نگه‌داری نام گروه و files مربوطه مورد استفاده قرار می‌گیرد که در زیر جدولی از دستورات معرفی شده است.

مشخصه	نوع	پیشفرض	توضیحات
name	رشته	بر فرض اینکه متغیر Group برابر x باشد، که در آن x یک عدد منحصر به فرد در میان تمام گروه‌ها در محصول (Product) خواهد بود.	نام گروه را نگه می‌دارد و برای خارج از آن قابل استفاده نیست.
files	لیست	لیست خالی	فایل‌های مورد نیاز را در پروژه معرفی می‌کند که می‌تواند برای اعمال مسیر و فیلتر سازی فایل‌ها تحت گروه و دیگر شاخه‌ها مورد استفاده قرار گیرد.
prefix	رشته	رشته خالی	توسط اعمال مقدار با اسلش در این متغیر می‌توان مسیر قبل از آن را مشخص کرد.
fileTagsFilter	لیست	لیست خالی	تگ مربوط به جستجو در رابطه با فایلهایی که مطابقت دارند را اعمال می‌کند. و هر نوع تغییرات بر نوع محصولات تحت تاثیر برچسب‌ها اعمال خواهد شد.
condition	بولین	صحیح / درست (true)	این مشخصه تعیین می‌کند کهایا واقعاً فایل مورد نظر بخشی از پروژه خواهد بود یا خیر، که بر اساس نوع شرط می‌توان آن را سفارشی سازی کرد.
fileTags	لیست	لیست خالی	برچسب‌ها برای ضمیمه شدن به فایلهای مربوط به گروه هستند. این‌ها می‌توانند توسط قوانین مربوطه معین شوند و بخشی از فایل‌ها را فیلتر کنند.
overrideTags	بولین	صحیح / درست (true)	تعیین می‌کند که چگونه فایلهایی که در سطح بالای پروژه و یا در داخل گروه خود به عنوان والد برچسب زده شوند.
excludeFiles	لیست	لیست خالی	مورد استفاده در نویسه عام، فایلهایی که در این لیست هستند به عنوان فایلی که باید در نظر گرفته نشود (حذف شود) اعمال خواهد شد.

در ادامه لیستی از پرچم‌های مورد استفاده قرار دارد که توسط آن‌ها می‌توان دستورات شرطی و نوع کامپایل برنامه را کنترل کرد.

مشخصه	نوع	پیشفرض	توضیحات
buildVariant	رشته	"debug" نوع اشکال زدایی - دیباگ	نام نوع حالت ساخت پروژه یا همان کامپایل را نگه می‌دارد که تحت مقادیر (release) و (debug) قابل پیکربندی است.
hostOS	لیست رشتہ‌ای	platform-dependent وابسته به پلتفرم	سیستم‌عاملی را تعیین می‌کند که قرار است برنامه بر روی آن کامپایل شود. "windows", "linux", "macos", "darwin", "unix", ...  توجه: دقت کنید که این مشخصه را با مشخصه qbs.targetOS اشتباه نگیرید چرا که این مقدار مشخص می‌کند برنامه بر روی چه سیستم‌عاملی قرار است اجرا شود.
targetOS	لیست رشتہ‌ای	platform-dependent وابسته به پلتفرم	سیستم‌عاملی را تعیین می‌کند که قرار است برنامه بر روی آن اجرا شود. مانند "windows", "linux", "macos", "darwin", "unix", "ios", "android", ... و غیره... "blackberry", "qnx", ...

برای مثال گروه زیر نحوه استفاده از مقادیر فوق را مشخص می‌کند که فقط فایل debughelper.cpp در حالت debug کامپایل شود:

```
Group {
    condition: qbs.buildVariant == "debug"
    files: "debughelper.cpp"
}
```

امکان این وجود دارد که ما بتوانیم تحت دستورات ویژه آن را اعمال کنیم:

```
Properties {
    condition: qbs.buildVariant == "debug"
    .cppdebugInformation: true
    .cppoptimization: "none"
}
```

و حتی امکان استفاده از این مشخصه در قالب QML فراهم می‌شود:

```
.cppdebugInformation: qbs.buildVariant == "debug" ? true : false
.cppoptimization: qbs.buildVariant == "debug" ? "none" : "fast"
```

در ادامه لیست مازول‌های قابل پشتیبانی در QBS آمده است:

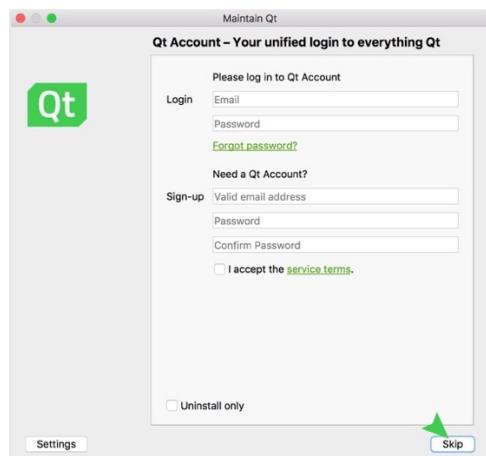
کتابخانه‌های بومی اندروید را برای ساخت فراهم می‌کند.	Module Android.ndk
پکیج‌های ساخت برای برنامه‌های اندروید را فراهم می‌کند.	Module Android.sdk
آرشیوهای مربوط به ساخت برنامه را فراهم می‌کند.	Module archiver
پشتیبانی از هسته‌های بنیادی بسته نرم‌افزاری را فراهم می‌کند.	Module bundle

پشتیبانی از C و C++ را فراهم می‌کند.	Module cpp
پشتیبانی از رابط ساخت و ابزارهای مرتبط و انواع فایل را برای اپل فراهم می‌کند.	Module ib
پشتیبانی از راهانداز Inno را فراهم می‌کند.	Module innosetup
پشتیبانی از Java را فراهم می‌کند.	Module java
پشتیبانی از ابزارهای lex و yacc را فراهم می‌کند.	Module lex_yacc
پشتیبانی از ماژول Nodejs را فراهم می‌کند.	Module nodejs
پشتیبانی از اسکریپت سیستم نصب کننده Nullsoft را فراهم می‌کند.	Module nsis
خصوصیات کلی و عمومی را برای Qbs فراهم می‌کند.	Module qbs
پشتیبانی از ساخت برنامه‌های QNX تحت QNX SDK را فراهم می‌کند.	Module qnx
پشتیبانی از TypeScript را فراهم می‌کند.	Module typescript
پشتیبانی از ابزارهای نصب ویندوز در قالب XML را فراهم می‌کند.	Module wix
پشتیبانی از Xcode را فراهم می‌کند.	Module xcode
پشتیبانی از Qt را فراهم می‌کند.	Qt Modules

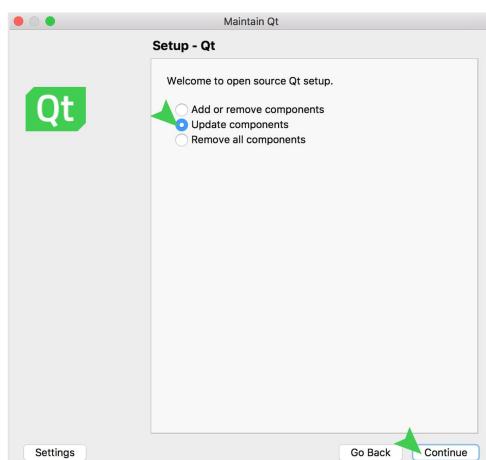
کیوبس در نسخه‌های ۵.۱۰ و ۶ کیوت با شماره ۱.۹.۰ فراهم خواهد شد که نسخه ۱.۸.۰ همینک در نسخه ۵.۹ قابل استفاده است.

## به روز رسانی کیوت بدون دریافت فایل نصبی آفلاین

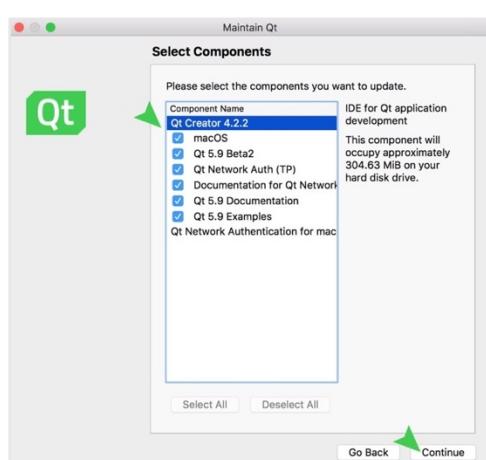
یکی از مشکلاتی که ممکن است بسیاری از ما با آن راحت نیستیم دریافت مجدد فایل و به روز رسانی آن به طور دستی با نصب آن است اما روشی است برای به روز رسانی آنلاین فایل‌های نصب شده بر روی سیستم که در ادامه به آن اشاره می‌کنیم. همانطور که می‌دانید کیوت چند سکویی است، بنابراین بر روی هر سکویی که هستید به مسیر نصبی آن رفته و فایل اجرایی Maintenance Tool را اجرا کنید در صورتی که حساب کاربری دارد آن را وارد کنید در غیر اینصورت گزینه Skip را انتخاب کنید و ادامه دهید:



در مرحله بعد برای به روز رسانی کافی است گزینه Update components را انتخاب و ادامه دهید.



در صورتی که کیوت نصب شده بر روی سیستم شما شامل آخرین ویرایش موجود بر روی مخازن qt.io نباشد گزینه‌های جدیدی برای به روز رسانی مشاهده خواهید کرد که می‌توانید با انتخاب آن‌ها و ادامه مراحل کیوت خود رو به روز رسانی نمایید.



### پشتیبانی بلند مدت

نسخه ۵.۹ یکی دیگر از نسخه‌هایی می‌باشد که به عنوان (LTS) پشتیبانی بلند مدت را فراهم آورده است. انتشار نسخه قبلی LTS با شماره ۵.۶ شروعی برای نمایش اولیه از پشتیبانی نسخه ۵ آن بود. برخی از تغییراتی عمدت‌های که در نسخه کیوت ۵.۶ وجود داشتند شامل بیش از ۲۰۰۰ اشکال بوده است که همه آن‌ها برای همیشه در کیوت رفع شده‌اند. علاوه بر آن، کیوت همین‌الآن از نسخه ۱۱ زبان قدرتمند C++ به طور کامل همراه با کامپایلرهای آن پشتیبانی می‌کند که اجازه داده است کدهای کیوت نو سازی شوند. همچنین سیستم Qt Lite به عنوان سیستم پیکربندی بر آن اضافه شده است که به روز رسانی‌های قابل توجهی را به معماری گرافیکی آن افزوده است. علاوه بر این موارد که اشاره شده است سایر موارد جدید که در نسخه ۵.۹ اجرا شده‌اند را در ادامه به آنها اشاره می‌کنیم.

### به عنوان یک نسخه LTS کیوت ۵.۹ برای سه سال آینده پشتیبانی خواهد شد

با پیشرفت‌های توسعه در CI و انتشارات در زیر ساخت‌ها توسعه و به روز رسانی‌ها و اصلاحیات کیوت ۵.۹ برای کاربران بسیار راحت و بیشتر خواهد بود. که برای سال اول به روز رسانی‌های مکرری صورت خواهد گرفت و سپس بعد از طی یک سال تغییرات و به روز رسانی‌ها بیشتر به عنوان برج (انشعاب)‌های مشخص صورت خواهند گرفت. بنابراین به عنوان یک نتیجه، انتظار می‌رود که سطح انتشار اصلاحیه‌ها و به روز رسانی‌ها در سال‌های دوم و سوم نسبت به سال اول کمتر شود. وعده‌اینکه کیوت ۵.۶ همین‌گونه پشتیبانی خواهد شد داده شده است، اما توسعه دهنده‌گان کیوت پیشنهاد می‌کنند که پروژه‌ها را بر پایه نسخه جدید یعنی ۵.۹ ایجاد و توسعه دهیم چرا که بهبودها و ویژگی‌های بسیار زیادی در نسخه ۵.۹ ارائه شده است که مزیت‌های بسیار بزرگی را برای شما در بر خواهند داشت.

نسخه ۵.۱۰ کیوت با مجهز شدن به ۸ لگان سبک و سیاق جدید و تجربه متفاوتی را با خود همراه خواهد ساخت که پیش نمایشی از نسخه ۶ کیوت خواهد بود. البته انتظار می‌رود که در نسخه ۶ کیوت ویژگی‌ها و حتی مازول و کلاس‌های مفید بسیاری اضافه شود و حتی با تغییرات قابل چشمگیری در محیط توسعه آن روبرو شویم. از این رو در این کتاب سعی شده است تنها در رابطه با موارد پر کاربرد تا نسخه ۵.۹ اشاره شود بنابراین نگران این نباشید که شاید موردی در این کتاب وجود داشته باشد که در نسخه ۶ کیوت سازگاری نداشته باشد چرا که بر اساس اطلاعیه‌های قبلی از مطرح کردن تمامی مواردی که قرار بوده‌اند منسخ شوند صرف نظر کرده‌ایم.

### چشم‌انداز فنی برای کیوت ۶

این چشم‌انداز احتمالاً برای دوست‌داران کتابخانه قدرتمند Qt و طرفدارانش جذاب باشد! بنابراین من سعی کرده‌ام تا نتایج پست رسمی کیوت را در رابطه با چشم‌انداز فنی برای آینده کیوت نسخه ۶ است در اختیار شما قرار دهم. تقریباً ۷ سال پیش کیوت نسخه ۵.۰ منتشر شد! از آن زمان بسیاری از چیزها در دنیای اطراف ما تغییر پیدا کرده است. و اکنون وقت آن رسیده است که چشم‌انداز جدیدی را از نسخه ۶ جدیدتر تعریف کنیم. بنابراین در این پست ما به معرفی مهمترین مواردی که به کیوت ۶ مرتبط است را می‌پردازیم.

به نقل از مدیر فنی کیوت Lars Knoll، کیوت ۶ دقیقاً ادامه کارهایی است که در نسخه ۵ انجام داده شده است. بنابراین توسعه باید به گونه‌ای باشد که کاربران نباید اذیت شوند. اما نسخه جدید می‌تواند یک آزادی بالاتر را در اجرای ویژگی‌های جدید، عملکرد و پشتیبانی بهتر از شرایط امروز و فردا از آنچه در حال حاضر می‌توان در سری ۵ داشته باشیم را به ما خواهد داد. همانطور که جزئیات بیشتر در زیر شرح داده شده است، کیوت ۶

هدف زیادی از سازگاری با نسخه قبلی خود یعنی کیوت ۵ را خواهد داشت. همچنین ما در حال توسعه روی نسخه ۶ نیز هستیم که قصد داریم برخی از ویژگی‌های کیوت ۶ را در نسخه‌های کیوت ۴ و کیوت ۵ LTS معرفی کنیم.

بنابراین با ثابت نگهداشتن ویژگی‌ها در کیوت ۱۴، بیشتر تمرکز تحقیق و توسعه به سمت کیوت ۶ تغییر خواهد یافت. بنابراین انتظار می‌رود کیوت ۶ تا پایان سال ۲۰۲۰ آماده شود. قبل از اینکه همه به چیزهای جدید بپردازیم، بباید برخی از ارزش‌های پایه از هسته اصلی کیوت را برای کاربران خود یادآوری کنیم تا چیزهایی که نمی‌خواهیم تغییر کنند را تعریف کنیم.

چه چیزی Qt را برای کاربران ما ارزشمند می‌کند؟

کیوت محصولی است که در بازارهای مختلفی مورد استفاده قرار می‌گیرد، ارزش‌های اصلی در هسته کیوت برای مشتریان و کاربران ما عبارتند از:

- ماهیت چند-سکویی آن، به کاربران این امکان را می‌دهد تا برنامه‌های خود را با استفاده از این فناوری به تمامی سیستم‌عامل‌های رو میزی، موبایل و سیستم‌های تعبیه شده (امیدها) مستقر کنند.
- مقایس پذیری آن از دستگاه‌های کم مصرف و یک منظوره تا برنامه‌های دسکتاب پیچیده و یا سیستم‌های متصل شده.
- رابطه‌های برنامه‌نویسی و ابزارها و مستندات در سطح جهانی، ایجاد برنامه‌ها را ساده‌تر می‌کند.
- حفظ، ثبات (پایداری) و سازگاری، امکان حفظ بانک بزرگی از کدها با حداقل تلاش برای نگهداری آن‌ها.
- یک اکو سیستم بزرگ توسعه‌دهنده با بیش از ۱ میلیون کاربر.

یک نسخه جدید از کیوت باید خواسته‌های محصول ما را مطابق با نیازهای بازار تنظیم کند، و در عین حال پنج ویژگی بالا را به خوبی حفظ کند. بازار دسکتاب، ریشه پیشنهادات و یک بازار قوی و مهم برای کیوت است؛ در این مرحله است که بیشترین تماس‌ها با ما و در انجمان‌های کیوت از طرف کاربران صورت می‌گیرد که باید سالم نگهداشت و رشد آن مهم باشد. بزرگترین بخش از رشد کیوت نیز مربوط به دستگاه‌های تعبیه شده و متصل شده می‌باشد؛ صفحات نمایش لمسی به تعداد تصاعدی در حال افزایش است که در کنار آن افزایش قیمت سخت‌افزار برای این دستگاه‌ها وجود دارد. چیزیستهای کم مصرف، میکروکنترلرهای همراه با صفحه نمایش لمسی به اندازه‌های کوچک در همه جا استفاده می‌شوند.

بسیاری از این دستگاه‌ها عملکردی نسبتاً ساده‌ای دارند، اما به رابط کاربری صیقلی و صافی نیاز دارند. بنابراین حجم زیادی از این دستگاه‌ها ایجاد می‌شود و ما باید اطمینان حاصل کنیم که می‌توانیم با ارائه خود آن فضا را هدف قرار دهیم تا بتوانیم مقایس پذیری خود را عملی کنیم. در عین حال، رابطه‌های کاربری در طیف دستگاه‌ها همچنان به افزایش پیچیدگی ادامه می‌دهند که شامل هزاران صفحه مختلف و برنامه‌های بسیاری است. ادغام عناصر سه بعدی و دو بعدی در یک رابط کاربری مشترک خواهد بود که در کنار آن استفاده از واقعیت افزوده و مجازی نیز وجود خواهد داشت.

عناصر هوش مصنوعی بیشتر در حوزه برنامه‌ها و دستگاه‌ها مورد استفاده قرار می‌گیرد و ما نیاز به روش‌های آنسان برای ادغام با آن‌ها داریم. رشد شدید تعداد دستگاه‌های متصل به هم و همچنین الزامات بسیار بالاتر در تجربه کاربر باعث می‌شود تا برای ساده سازی ایجاد برنامه‌ها و دستگاه‌ها، روی ابزارهای کلاس جهانی تمرکز کنیم. هماهنگ سازی و ادغام طراحان UX در گرددش کار توسعه یکی از اهداف است؛ اما بسیاری از زمینه‌های دیگر وجود خواهد داشت که ما باید برای ساده سازی زندگی کاربران تلاش کنیم. کیوت ۶ یک نسخه اصلی و جدید برای Qt خواهد بود؛ هدف اصلی با چنین نسخه اصلی و جدید، آماده سازی کیوت برای شرایط مورد نیاز در سال ۲۰۲۰ و بعد از آن، تمیز کردن کدهای پایه ما و حفظ آسان‌تر است. به همین ترتیب تمرکز روی آن مواردی خواهد بود که نیاز به تغییرات معماري در کیوت دارند و بدون شکستن برخی از سازگاری‌ها با سری‌های کیوت ۵ قابل انجام نیست.

در زیر برخی از تغییرات اساسی که ما باید در کیوت ایجاد کنیم برای مناسب‌تر کردن آن برای سال‌های آینده ارائه شده است.

### نسل بعدی کیوام‌آل (QML)

زبان QML و فناوری‌های اصلی رشد سال‌های گذشته‌ما بوده است. روش‌های بصری ایجاد واسطه‌های کاربری با استفاده از آن فناوری‌ها نقطه فروش بی نظیری از پیشنهاد ما است. اما QML، همانطور که برای کیوت ۵ ایجاد شده است، دارای تعداد زیادی تغییرات ناگهانی و محدودیت است. این به نوبه خود به این معنا است که، امکان پیشرفت‌های چشمگیری وجود دارد که ما قصد داریم با کیوت ۶ آن‌ها را پیاده سازی کنیم.

معرفی وابستگی زیاد به نوع (strong typing)، وابستگی کم به نوع (weak typing) امکان ایجاد تغییر در کدها را برای کاربران ما سخت می‌کند. سیستمی از مدل وابستگی زیاد به نوع امکان پشتیبانی از این تغییرات را در محیط‌های یکپارچه توسعه نرم افزار و سایر ابزارها به کاربران می‌دهد و به طور چشمگیری حفظ و نگهداری از آن‌ها را راحت می‌کند. همچنین، قادر به تولید کدهای اجرایی هرچه بهتر و با سریار کمتر خواهیم بود.

اعمال JavaScript به عنوان یک ویژگی اختیاری، با توجه به این موضوع، داشتن یک موتور کامل جاوا اسکریپت هنگام استفاده از QML می‌تواند مشکلات را پیچیده‌تر کند و به خصوص هنگام هدف قرار دادن سخت‌افزار کم مصرف مانند میکرو کنترلرهای یک مشکل اصلی محسوب می‌شود. اما در بسیاری از موارد استفاده از آن بسیار مفید است.

حذف نسخه سازی QML، با ساده کردن برخی از قوانین بررسی و جستجو و تغییرات در برخی از خواص می‌توانیم نیاز به نسخه را در QML حذف کنیم. این به نوبه خود منجر به ساده سازی‌های زیاد در موتور کیوام‌آل می‌شود. حجم کار در حفظ فناوری کیوت کوئیک و ساده‌تر کردن استفاده از Qt Quick و QML را برای کاربران بسیار ساده‌تر خواهد کرد.

### حذف ساختار داده‌های تکراری بین QML و QObject

در حال حاضر، برخی از ساختار داده‌ها بین QML و meta-object کپی و تکرار می‌شوند و عملکرد (کارایی و پرفرمننس) را در استارت‌آپ برنامه کاهش می‌دهد و باعث افزایش مصرف حافظه نیز می‌گردد. بنابراین با متحده کردن ساختارهای داده‌ها، ما قادر خواهیم بود بخشی اعظمی از آن را حذف کنیم.

### خودداری کردن از ساختارهای داده تولید شده

این مربوط به نکته قبل است، جایی که در حال حاضر بسیاری از ساختارهای داده تکراری در زمان اجرا تولید می‌شوند. باید تولید اکثر آن‌ها در زمان کامپایل کاملاً امکان‌پذیر باشد.

پشتیبانی از کامپایل QML برای بهره‌وری از کدهای بومی C++، با وابستگی زیاد به نوع و قوانین جستجوی ساده‌تر، می‌توانیم QML را به کدهای بومی C++ تبدیل کنیم که نتیجه آن به طور قابل توجهی عملکرد زمان اجرا را افزایش می‌دهد.

پشتیبانی از پنهان کردن جزئیات اجرا، روش و خصوصیات «خصوصی» یک نیاز طولانی مدت است تا بتوانید داده‌ها و عملکردها را در اجزای QML پنهان کنید.

هماهنگ‌سازی و ادغام بهتر ابزارها، مدل کدهای ما غالباً برای QML ناقص است و باعث می‌شود که تغییر مکان و خطاهای را در زمان کامپایل غیر ممکن کند. با تغییرات فوق، می‌توان تشخیص کامپایلر را ارائه داد که بتواند با C++ و همچنین پشتیبانی از آن پالایش کدها را بهبود بخشد.

## نسل بعدی گرافیک‌ها

بسیاری از موارد در حوزه گرافیک در نسخه کیوت ۵ تغییر یافته‌اند. این باعث می‌شود که برای حفظ رقابت و توسعه در پُشته انجام شود. با کیوت ۵، ما از رابطهای برنامه‌نویسی OpenGL را برای گرافیک‌های ۳ بعدی استفاده کردیم. از آن زمان به بعد، میزبانی از رابطهای برنامه‌نویسی جدید نیز تعریف شده است. بنابراین ولکان (Vulkan) جانشین مشخصی برای OpenGL در لینوکس است، اپل نیز Metal (Metal) را تحت فشار قرار داد تا آن را جایگزین کند و مایکروسافت DirectX را دارد. این بدان معنی است که کیوت در آینده مجبور است به صورت یکپارچه با تمام رابطهای برنامه‌نویسی کار کند. برای اینکه این ویژگی امکان‌پذیر باشد، باید یک لایه جدید که رابطهای برنامه‌نویسی گرافیکی را انتزاع می‌کند مانند QPA (Qt Quick Scenegraph) به نام رابط سخت‌افزاری RHering تعریف شود. ما نیز باید زیر ساخت‌های ارائه شده خود (Qt Quick Scenegraph، Qt Quick Scenegraph و پشتیبانی ۳ بعدی) را در بالای آن لایه قرار دهیم.

مجموعه رابطهای برنامه‌نویسی گرافیکی مختلف باعث می‌شود که ما از زبان‌های مختلف سایه‌زنی پشتیبانی کنیم. ابزار Qt Shader به عنوان یک ماثول به ما کمک می‌کند تا سیستم سایه‌زنی را به صورت همزمان (کراس-کامپایل) و در زمان اجرا کامپایل کنیم. بحث ۳ بعدی نقش مهم و مهمتری را ایفا می‌کند، و پشتیبانی فعلی ما یک راه حل یکپارچه برای ایجاد رابط کاربری (UI) هایی که حاوی هر دو عنصر ۲ و ۳ بعدی باشد را ندارد. ادغام با محتوا از Qt 3D Studio و یا Qt3D QML با این همگام سازی انیمیشن‌ها و انتقال‌ها بر روی یک فریم با سطح فریم بین محتوای ۲ و ۳ بعدی غیر ممکن است.

ادغام جدید محتوای ۳ بعدی با فناوری کیوت کوئیک با هدف حل این مشکل ایجاد شده است. در این حالت، یک سیستم ساخت (رندر) کامل و جدید به شما امکان می‌دهد تا محتوای ۲ و ۳ بعدی را با هم ضبط کنید. با این کار QML به زبان UI تعریف و تبدیل می‌شود که سه بعدی هستند و نیاز به فرمت UIP برطرف می‌شود. ما یک پیش‌نمایش از کیوت کوئیک جدید با پشتیبانی سه بعدی در حال حاضر با کیوت ۵.۱۴ ارائه می‌دهیم که اطلاعات بیشتر در یک پست جداگانه ارائه خواهد شد.

سرانجام پُشته گرافیکی جدید نیاز به پشتیبانی از خط لوله برای چیزهای گرافیکی هستند که این امکان را می‌دهد تا آن‌هایی که در زمان کامپایل برای سخت‌افزار مورد نظر تهیه شده‌اند آماده کرده و از موارد مورد نظر استفاده کند. برای مثال، فایل‌های PNG را به بافت‌های فشرده تبدیل می‌کند و بسیاری از آن‌ها را به بافت (Texture) تبدیل کند. سایه‌ها و مشه‌ها را به قالب‌های باینری بهینه شده و موارد دیگر تبدیل خواهد کرد.

همچنین هدف ما این است که یک موتور متعدد برای پوسته/ظاهر در کیوت ۶ ارائه دهیم که به ما این امکان را می‌دهد تا از نظر ظاهری و احساسات بر روی دسکتاپ و موبایل آن را بر روی هر دو فناوری کیوت ویجت و کیوت کوئیک ارائه کنیم.

ابزارهای گرافیکی ما برای ساخت رابطهای کاربری به دو بخش با استودیو کیوت ۳ بعدی (Qt 3D Studio) و استودیو طراحی کیوت (DesignStudio تقسیم بندی شده‌اند

ابزارهای طراحی نیز به ایجاد محتوا مانند، محتوای ساخته شده در Photoshop ، Sketch ، Illustrator ، Maya ، ۳DsMax و دیگر موارد ادغام شده است. ابزارهای توسعه به توجه زیادی برای تمرکز دارد تا بتوانیم بهترین‌ها را در پشتیبانی کلاس برای QML ، C++ و پایتون ارائه دهیم. یک ابزار متعدد و یکپارچه این اجازه را می‌دهد که یک طراح UX بتواند از قابلیت‌های طراحی در کیوت کریتور استفاده کند و طراحان می‌توانند از ویژگی‌های ابزارهای توسعه‌دهنده مانند تهیه یک پروژه یا آزمایش روی یک دستگاه بهره‌مند شوند.

ابزار ساخت QMake به عنوان ابزار ساخت در کیوت ۵ مورد استفاده قرار می‌گیرد که تعداد زیادی تغییرات ناگهانی و محدودیت‌ها خواهد داشت. برای کیوت ۶، هدف ما این است که CMake را به عنوان سیستم ساخت ثالث و استاندارد برای ساخت خود کیوت استفاده کنیم. چرا که سی‌میک تاکنون پرکاربردترین سیستم ساخت در جهان برای سی‌پلاس‌پلاس بوده است و ادغام هرچه‌بهتر آن کاملاً مورد نیاز است. البته پشتیبانی از QMake ادامه خواهد داشت، اما آن را توسعه نخواهیم داد یا از آن برای ساخت فریمورک کیوت استفاده نخواهیم کرد.

### بهبود رابطهای برنامه‌نویسی C++

سی‌پلاس‌پلاس طی سال‌های گذشته تغییرات بسیار زیادی کرده است؛ در حالی که ما مجبور بودیم کیوت ۵.۰ را روی سی‌پلاس‌پلاس ۹۸ پایه‌گذاری کنیم. اما اکنون می‌توانیم به سی‌پلاس‌پلاس ۱۷ برای پایه‌گذاری کیوت ۶ اطمینان کنیم. این بدان معنی است که C++ عملکرد بسیار بیشتری را نسبت به زمان توسعه و اجرای کیوت ۵ که در دسترس نبود ارائه خواهد کرد. هدف ما با کیوت ۶ بهتر شدن با یکپارچه‌سازی و ادغام قابلیت‌ها بدون از دست دادن پشتیبانی و سازگاری از روش‌های پیشین (رو به عقب یا همان backward compatibility ) است.

برای کیوت ۶، هدف ما این است که برخی از قابلیت‌های معرفی شده با QML و فناوری Qt Quick را از طرف C++ در دسترس قرار دهیم. بنابراین ما در تلاش برای معرفی یک سیستم خاص برای QObject و کلاس‌های مرتبط هستیم. موتور اتصال دهنده را از QML در هسته کیوت ادغام می‌کنیم و آن را از سی‌پلاس‌پلاس در دسترس قرار می‌دهیم. این سیستم خاص از موتور اتصال به کاهش قابل توجهی در سریار زمان کار و مصرفه حافظه در اتصال منجر می‌شود و آن‌ها را برای همه قسمت‌های Qt Quick قابل دسترس می‌کند.

### پشتیبانی از زبان

با کیوت ۵.۱۲، پشتیبانی از پایتون معرفی شده است. همچنین مروگر را به عنوان پلتفرم جدید از طریق کیوت برای وب اسملی اضافه کرده‌ایم. پس از انتشار کیوت ۶.۰ نگه‌داشتن و گسترش بیشتر بر روی سطح چند-سکویی بخش مهمی از اهداف و مسیر توسعه سری‌های کیوت ۶ خواهد بود.

### سازگاری با کیوت ۵ و افزایش سازگاری‌ها و بهبودها

سازگاری با نسخه‌های قدیمی‌تر بسیار مهم است، بنابراین وقتی کیوت ۶ را توسعه می‌دهیم یک نیاز اساسی محسوب می‌شود. توسط چهارچوب کیوت میلیون‌ها خط کد نوشته شده است و هرگونه تغییرات در ناسازگاری که انجام شود هزینه‌ای را برای کاربران خواهد داشت. علاوه بر این، کار بیشتری برای تغییرات در کیوت ۶ نیاز است تا کاربران کم با آن سازگار شوند که منجر به هزینه‌های بیشتر از طرف تیم توسعه کیوت برای حفظ آخرین نسخه کیوت ۵ خواهد بود.

به این ترتیب، ما باید به فکر جلوگیری از ساطع شدن خطاهای احتمالی، در زمان کامپایل و یا در زمان اجرا باشیم. در حالی که ما نیاز به حذف بخش‌هایی از کیوت خواهیم داشت، باید اطمینان حاصل کنیم که کاربران ما از عملکرد مورد نیاز خود برخوردار هستند. این بدان معنا است که کلیدهایی مانند Qt Widgets و سایر قسمت‌هایی که توسط بخش بزرگی از کاربران ما مورد استفاده قرار می‌گیرد، در دسترس باشد.

ما در حال برنامه‌ریزی برای افزایش بسیاری از پیشرفت‌ها در کلاس‌های اصلی و عملکردی هستیم که در سری کیوت ۵ نتوانستیم انجام دهیم. هدف این است که سازگاری کامل منبع را حفظ کنیم، اما از آنجا که می‌توانیم سازگاری باینری را با کیوت ۶ بشکنیم، می‌توانیم پاک‌سازی‌ها و اصطلاحات کاملاً زیادی را انجام دهیم که در کیوت ۵ نمی‌توانستیم آن را عملی کنیم.

با این وجود، ما باید به جلو پیش برویم و برخی از پاک‌سازی‌ها که در کیوت ۵ در مورد کلاس‌ها، توابع و یا مارژول‌ها عنوان شده بود را در کیوت ۶ به طور کامل اعمال کنیم. این کار باعث می‌شود ما روی مبنای کدگذاری شده فعلی تمرکز بیشتر و بهتری داشته باشیم.

با این حال، انتقال به دور از قسمت‌های منسوخ شده باید تا حد امکان ساده باشد و کاربران ما می‌توانند با استفاده از کیوت ۵.۱۵ «پشتیبانی بلند مدت» به صورت ایده‌آل این کار را انجام دهند. هدف ما باید این باشد که کیوت ۶ به اندازه کافی با نسخه ۵.۱۵ سازگار باشد تا فرد بتواند به راحتی یک بخش اعظمی از کد خود را حفظ کند، به طوری که کد آن در هر دو نسخه ۵ و ۶ قابل کامپایل باشد.

## بازار و ساختار فنی محصول

علاوه بر بهبود چهارچوب و ابزارهای کیوت، هدف ما ایجاد بازار جدیدی برای قطعات و ابزارهای توسعه است. این بازار بر روی کاربران مستقیم ما متمرکز خواهد شد که برنامه‌های کاربردی و دستگاه‌های تعبیه شده را طراحی و توسعه می‌دهند؛ به این ترتیب این یک مرکز تجمع اصلی برای اکو سیستم کیوت خواهد بود که این امکان را به شخص ثالث می‌دهد تا نسخه‌های اضافی خود را در کیوت منتشر کنند و هم محتوای رایگان و تجاری را که برای آن هزینه پرداخت می‌کنند.

کیوت طی سال‌های گذشته رشد بسیار زیادی داشته است، تا جایی که ارائه نسخه جدید آن یک کار مهم است. با استفاده از کیوت ۶ فرصتی برای بازسازی محصولات ارائه شده ما وجود دارد و یک محصول اصلی و کوچکتر که شامل چهارچوب‌ها و ابزارهای اساسی است. ما از بازار استفاده خواهیم کرد تا چهارچوب و ابزارهای اضافی خود را ارائه دهیم، نه به عنوان یک بسته‌نرم‌افزاری وابسته به کیوت!

## پیشنهادات و ملاحظات در عملکرد و کارآیی

همانطور که می‌دانید تولید و توسعه یک نرم‌افزار با سرعت و کارآیی بالا یکی از مزایای توسعه‌دهنگان است که به تنها یک می‌تواند به عنوان یک فاکتور بسیار مهم مطرح شود. بنابراین در صورتی که شما محصولی را در قالب وب، موبایل یا دسکتاپ تولید می‌کنید باید به آن توجه داشته باشید که سرعت اجرایی برنامه شما در زمان استارت‌اپ و بعد در مراحل پردازشی و فرایندهای دیگر باید مورد قبول باشد.

در این میان برنامه‌نویسان سی++ می‌دانند که باید در مدیریت حافظه و دیگر موارد با دقت بیشتری عمل کنند تا بتوانند به حداقل نشتنی در حافظه (حتی فاقد نشتنی در آن) و در نهایت رسیدن به حداقل سرعت پردازشی توجه داشته باشند. با توجه به این موضوع که کتابخانه Qt به عنوان یکی از کتابخانه‌های این زبان بشمار می‌آید؛ بسیاری از توسعه‌دهنگان درگیر توسعه بخش فرانت‌اند نیز می‌شوند. بنابراین باید توجه داشت در زمان توسعه رابط کاربری مبتنی بر فناوری Qt Quick مواردی را جهت افزایش کارآیی سرعت و هماهنگی کامل با سی++ را در نظر بگیرند چرا که بدون در نظر گرفتن فاکتورهای زمانی نتیجه آن بسیار بد خواهد بود.

شما به عنوان یک توسعه‌دهنده نرم‌افزار، باید برای رسیدن حداقل فریم ریت ۶۰ در موتور رندرینگ تلاش کنید. این میزان بدین معنی است که بین هر فریم که در میان آن‌ها می‌تواند پردازش انجام شود، تقریباً ۱۶ میلی ثانیه وجود دارد که شامل پردازش مورد نیاز برای بارگذاری‌های اولیه به سمت سخت‌افزار گرافیکی می‌باشد.

در عمل، این به این معنی است که توسعه‌دهنده نرم‌افزار باید:

- هر کجا که ممکن است از روش ناهمزن (Asynchronous) در بخش برنامه‌نویسی رویدادمحور (Event-driven programming) استفاده کند.
- در بخش‌هایی که به میزان قابل توجهی شامل پردازش می‌شوند از وُرکر ترد (worker threads) استفاده کند.
- هرگز حلقه رویداد را به صورت دستی برای چرخش و ادامه تنظیم نکند.
- هرگز بیش از چند میلی ثانیه برای هر فریم در بلوک توابع صرف نکند.

عدم انجام این کارها باعث پرسش در فریم‌ها می‌شود که تاثیر بسیار جدی در تجربه کاربری دارد.

نکته: الگویی که وسوسه انگیز است اما نباید هرگز از آن استفاده شود، چرا که یک **QEventLoop** و یا صدا زدن به منظور جلوگیری از مسدود شدن در یک بلوک گُد سی++ که در QML استفاده شده است به کار گرفته می‌شود. این خطرناک است، زیرا زمانی که یک حلقه رویداد در هندرل سیگنال و یا پیوند (Binding) وارد می‌شود، موتور QML همچنان برای اجرای سایر پیوندها، اینیمیشن‌ها، ترنسیشن‌ها (تبديلات) و غیره می‌پردازد. این اتصالات می‌توانند باعث عوارض‌های جانبی شوند.

## پروفایلینگ (Profiling)

مهمترین نکته این است که: از ابزار **QML Profiler** که بخشی از محیط توسعه یکپارچه نرم‌افزار **Qt Creator** است استفاده کنید. زیرا دانستن زمانی که در یک برنامه صرف می‌شود به شما امکان این را می‌دهد تا بر قسمت یا بخشی که در آن مشکل وجود دارد تمرکز سریع‌تر و بهتری داشته باشید. برای نحوه استفاده از این ابزار به کتابچه راهنمای خود کیوت کریتور مراجعه کنید.

تعیین کردن اینکه اغلب کدام پیوند (اتصال) در حال اجرا است، یا کدام یک از توابع شما بیشترین زمان را برای اجرا صرف می‌کنند، به شما این امکان را می‌دهد که تصمیم بگیرید که آیا شما نیاز به بهینه سازی آن بخش از گُدها را دارید یا نیاز خواهد بود آن بخش از کد خود را مجدداً پیاده سازی کنید تا عملکرد آن بهتر شود یا خیر. به طور کلی تلاش برای بهینه سازی بدون استفاده از ابزار QML Profiler احتمالاً باعث بهبودهای جزئی و غیرقابل توجه در عملکرد خواهد شد.

## گُد جاوا اسکریپت (JavaScript)

بیشتر برنامه‌های QML مقدار زیادی از کدهای جاوا اسکریپت در درون خود خواهند داشت. که به شکل توابع پویا، مدیریت سیگنال و یا عبارت‌های مربوط به پراپرتی‌ها (مشخصه، اموال) را در اختیار دارند. به طور کلی این یک مشکل نیست. با تشکر از برخی از بهینه سازی‌های موتور QML، مانند آنهایی که به کامپایلر منتقل می‌شوند، می‌تواند (در بعضی موارد استفاده شده) حتی سریع‌تر از فراخوانی‌های سمت سی++ باشد. با این حال، باید اطمینان حاصل شود که پردازش غیر ضروری به طور تصادفی اتفاق نیفتاده باشد.

### اتصالات (Binding)

دو نوع اتصال در QML وجود دارد: اتصالات بهینه شده و غیر بهینه شده. ایدهٔ خوبی است که عبارات مربوطه را به همان اندازه ساده‌تر حفظ کنید. زیرا موتور QML برای ارزیابی عبارات از یک ارزیابی ساده و بهینه شده استفاده می‌کند که می‌تواند مشخص کند تا نیازی به تغییر آن در محیط جاوا اسکریپت نباشد. این اتصالات بهینه سازی شده بسیار کارآمدتر از اتصالات پیچیده‌تر (غیر بهینه سازی شده) هستند. الزامات اساسی نیز برای این گونه اتصال‌ها این است که اطلاعات نوع هر نماد که به آن دسترسی دارد باید در زمان کامپایل شناخته شده باشد.

عواملی که باعث جلوگیری از به حداقل رساندن بهینه سازی در عبارات اتصالات می‌شود:

- تعریف متغیرهای واسط توسط جاوا اسکریپت
- دسترسی به خواص var
- صدا زدن توابع جاوا اسکریپت
- ساخت (آغاز یا خاتمه یافتن) و یا تعریف توابع در میان عبارت اتصالات
- دسترسی به خواص (پراپرتی‌های) خارج از محدوده ارزیابی فوری
- نوشتن در مورد سایر عوامل به عنوان عوارض جانبی

توجه داشته باشید، زمانی اتصالات سریع‌تر می‌شوند که نوع اشیاء و خواصی که با آنها کار می‌کنند مشخص شده باشند. این به این معنی است که، خواص غیرنهایی (non-final) ممکن است در بعضی موارد کُندرت باشد. (جایی که ممکن است یک خاصیت تغییر کرده باشد).

محدوده‌هایی ای که برای ارزیابی فوری می‌توان در نظر گرفت قسمت زیر هستند:

- ارزیابی در خاصیت عبارت‌های دامنه یک شیء (برای عبارت‌های اتصال، این است که شیء به کدام یک از خاصیت‌های اتصال تعلق دارد).
- ارزیابی در شناسه‌های هر یک از اشیاء موجود در کامپوننت (جزء)
- ارزیابی در خاصیت‌های ریشه‌آیتم در یک کامپوننت (جزء)

شناسه‌های اشیاء از دیگر کامپوننت‌ها و خاصیت‌های هر یک از اشیاء، مانند نمادهای تعریف شده و یا وارد شده از طرف جاوا اسکریپت در محدوده ارزیابی فوری نیستند. به این ترتیب اتصالاتی که به این موارد مربوط هستند نمی‌توانند بهینه سازی شوند.

نکته: توجه داشته باشید که اگر نمی‌توانید اتصالات مورد نظر خود را توسط موتور QML بهینه سازی کنید، در این صورت باید آن را در محیط کامل جاوا اسکریپت بهینه سازی نمایید.

یک مزیت عمدی برای استفاده از جاوااسکریپت این است که در اغلب موارد، هنگامی که به یک ویژگی از نوع QML دسترسی پیدا می‌شود، یک شیء جاوااسکریپت با منبع حاوی سی‌پلاس‌پلاس پایه (یا یک مرجع آن) ایجاد می‌گردد. در بیشتر موارد، این عمل به نسبت کم‌هزینه است، اما در سایر موارد می‌تواند بسیار پرهزینه باشد. یک مثال از پرهزینه بودن آن، ارجاع کردن **Q\_PROPERTY** و **QVariantMap** به ویژگی نوع QML است. لیست‌ها نیز می‌توانند هزینه بالایی داشته باشند. اگرچه دنباله ا نوع خاص (QList، int، qreal، bool، QString و QUrl) باید کم‌هزینه باشد. تبدیل انواع دیگر لیست هزینه زیادی دارد (ایجاد آرایه جدید جاوااسکریپت و افزودن تک به تک انواع جدید، با تبدیل هر نوع از نمونه نوع سی‌پلاس‌پلاس به مقدار جاوااسکریپت).

## رفع سازی خواص‌ها

تفکیک پذیری در خواص (پرپرتبهای) زمان بر است. در حالی که در بعضی از موارد نتیجه آن می‌تواند ذخیره شده و دوباره مورد استفاده قرار بگیرد. بهتر است همیشه از انجام کارهای غیر ضروری جلوگیری شود. در مثال زیر، ما یک بلوک کد داریم که اغلب اجرا می‌شود (در این مورد، آن شامل یک حلقه صریح است؛ اما برای مثال می‌توان آن را با یک عبارت پیوند و مورد ارزیابی قرار داد). در این مثال مشکل را با در نظر گرفتن شناسه "rect" و خاصیت رنگ "color" آن را چندین بار تغییر و تولید می‌کنیم.

این مثال کاری که می‌خواهیم را انجام می‌دهد، اما روش بهینه شده ای نیست بنابراین مثال زیر بسیار بد خواهد بود:

```
import QtQuick 2.3

Item {
    width: 400
    height: 200
    Rectangle {
        id: rect
        anchors.fill: parent
        color: "blue"
    }

    function printValue(which, value) {
        console.log(which + " = " + value);
    }

    Component.onCompleted: {
        var t0 = new Date();
        for (var i = 0; i < 1000; ++i) {
            printValue("red", rect.color.r);
            printValue("green", rect.color.g);
            printValue("blue", rect.color.b);
            printValue("alpha", rect.color.a);
        }
        var t1 = new Date();
        console.log("Took: " + (t1.valueOf() - t0.valueOf()) + " milliseconds for
1000 iterations");
    }
}
```

برای اینکه کد فوق بهینه سازی شود بهتر است به صورت زیر باشد:

```
import QtQuick 2.3
```

```

Item {
    width: 400
    height: 200
    Rectangle {
        id: rect
        anchors.fill: parent
        color: "blue"
    }

    function printValue(which, value) {
        console.log(which + " = " + value);
    }

    Component.onCompleted: {
        var t0 = new Date();
        for (var i = 0; i < 1000; ++i) {
            var rectColor = rect.color; // resolve the common base.
            printValue("red", rectColor.r);
            printValue("green", rectColor.g);
            printValue("blue", rectColor.b);
            printValue("alpha", rectColor.a);
        }
        var t1 = new Date();
        console.log("Took: " + (t1.valueOf() - t0.valueOf()) + " milliseconds for
1000 iterations");
    }
}

```

فقط این تغییر ساده باعث بهبود عملکرد قابل توجهی می‌شود. بنابراین، توجه داشته باشید که کد بالا را می‌توان حتی بهبود بیشتری داد (از آنجا که ویژگی مورد نظر هرگز در طول پردازش حلقه تغییر نکرده است)، با بالا بردن دقت خارج از حلقه، به صورت زیر است:

```

import QtQuick 2.3

Item {
    width: 400
    height: 200
    Rectangle {
        id: rect
        anchors.fill: parent
        color: "blue"
    }

    function printValue(which, value) {
        console.log(which + " = " + value);
    }

    Component.onCompleted: {
        var t0 = new Date();
        var rectColor = rect.color; // resolve the common base outside the tight
loop.
        for (var i = 0; i < 1000; ++i) {
            printValue("red", rectColor.r);
            printValue("green", rectColor.g);
            printValue("blue", rectColor.b);
            printValue("alpha", rectColor.a);
        }
        var t1 = new Date();
        console.log("Took: " + (t1.valueOf() - t0.valueOf()) + " milliseconds for
1000 iterations");
    }
}

```

## پیوند خاصیت‌ها (Property Binding)

در صورتی که خواص مرجع هر یک از اتصالات تغییر یابند، به آن خاصیت متصل دوباره ارزیابی خواهد شد. به همین ترتیب، عبارات خواص باید به همان اندازه ساده باشند. اگر شما یک حلقه دارید که در آن پردازشی را انجام می‌دهید، اما تنها نتیجه نهایی پردازش مهم است، اغلب بهتر است که موقتاً آن را بهروز کرده و در صورت نیاز آن را برای بهروز رسانی یک خاصیت مورد نیاز اختصاص دهید. این کار به منظور جلوگیری از ارزیابی‌های مجدد بسیار مفید خواهد بود.

مثال ذکر شده زیر این نکته را نشان می‌دهد که بسیار بد پیاده سازی شده است:

```
import QtQuick 2.3

Item {
    id: root
    width: 200
    height: 200
    property int accumulatedValue: 0

    Text {
        anchors.fill: parent
        text: root.accumulatedValue.toString()
        onTextChanged: console.log("text binding re-evaluated")
    }

    Component.onCompleted: {
        var someData = [ 1, 2, 3, 4, 5, 20 ];
        for (var i = 0; i < someData.length; ++i) {
            accumulatedValue = accumulatedValue + someData[i];
        }
    }
}
```

حلقه‌ای که در رویداد **onCompleted** می‌باشد، موجب می‌شود تا خاصیت متن **text** به تعداد ۶ بار مجدداً مورد ارزیابی قرار بگیرد (که در نتیجه هر گونه پیوند مرتبط با مقدار متن ممکن است) همچنین کنترلر سیگنال **onTextChanged** هر یک را مجدداً ارزیابی می‌کند و این موجب می‌شود متن هر بار نمایش داده شود. این امر ضروری نیست چرا که هدف ما دسترسی به ارزش نهایی آن خواهد بود.

روش پیشنهادی به صورت زیر خواهد بود:

```
import QtQuick 2.3

Item {
    id: root
    width: 200
    height: 200
    property int accumulatedValue: 0

    Text {
        anchors.fill: parent
        text: root.accumulatedValue.toString()
        onTextChanged: console.log("text binding re-evaluated")
    }
}
```

```

        }

Component.onCompleted: {
    var someData = [ 1, 2, 3, 4, 5, 20 ];
    var temp = accumulatedValue;
    for (var i = 0; i < someData.length; ++i) {
        temp = temp + someData[i];
    }
    accumulatedValue = temp;
}
}

```

## بارگذاری‌ها و منابع مرتبط به تصاویر

تصاویر بخش مهمی از رابط کاربری هستند. متأسفانه، به دلیل زمان بارگیری آنها، میزان حافظه ای که مصرف می‌کنند نیز قابل توجه است. بنابراین توجه داشته باشید که در صورتی که اپلیکیشن شما شامل تصاویری با اندازه بزرگتر هستند، اما شما آن را در اندازه کوچکتر از واقعی خود نشان می‌دهید. در این صورت بهتر است اندازه سورس آن را نیز مشخص کنید که در خاصیت "sourceSize" مشخص می‌شود. این کار باعث می‌شود تا مطمئن شویم عنصر تولید شده با مقیاس کوچکتری در حافظه نگهداری می‌شود. البته مراقب این نیز باشید که تغییر اندازه سورس موجب می‌شود تصویر دوباره بارگیری شود.

## بارگیری غیرهمزمان (Asynchronous)

معمولًاً برنامه‌های زیبا و با کیفیت دارای تصاویر با کیفیت نیز هستند، بنابراین لازم است تا اطمینان حاصل شود که بارگیری یک تصویر یک رشته از UI را بلوک یا مسدود نمی‌کند. خاصیت **asynchronous** در نوع **QML Image** را با مقدار **true** تنظیم کنید تا بارگیری غیرهمزمان از تصاویر بر روی سیستم فایل‌های محلی بارگیری شود. این کار از وارد کردن فشار بر روی تصاویر و رابط کاربری جلوگیری خواهد کرد که نتیجه آن حفظ زیبایی رابط کاربری شما در زمان بارگذاری تصاویر خواهد شد.

## نکات ریز اما مهم

- از سایه پردازی‌ها و تصاویری که در حین زمان اجرا در ترکیب تصاویر شما ایجاد می‌شوند دوری کنید.
- توجه داشته باشید که در صورت عدم نیاز از خاصیت **smooth** استفاده نکنید. فعال سازی این خاصیت موجب تولید تصاویر صاف و با کیفیت می‌شود که در سخت افزارهای قدرتمند توصیه می‌شود. در سخت افزارهای ضعیف تولید تصاویر صاف زمان برخواهد بود و در صورتی که اندازه تصویر واقعی نباشد بر روی نتیجه آن تاثیری نخواهد داشت.
- از نقاشی و یا رسم طرح بر روی یک ناحیه به صورت پشت سر هم اجتناب کنید. از نوع **Item** به عنوان عنصر ریشه به جای **Rectangle** استفاده کنید.

## عناصر موقعیت با لنگر

استفاده از لنگرهای (anchors) به جای اتصالات (bindings) به یک آیتم موقعیت نسبتاً تأثیر بیشتری خواهد داشت. برای مثال برای اتصال موقعیت

rect1 را توجه کنید:

```

Rectangle {
    id: rect1
    x: 20
    width: 200; height: 200
}
Rectangle {
    id: rect2
    x: rect1.x
    y: rect1.y + rect1.height
    width: rect1.width - 20
    height: 200
}

}

```

این کار را می‌توان توسط لنگرها به طور بسیار موثری انجام داد:

```

Rectangle {
    id: rect1
    x: 20
    width: 200; height: 200
}
Rectangle {
    id: rect2
    height: 200
    anchors.left: rect1.left
    anchors.top: rect1.bottom
    anchors.right: rect1.right
    anchors.rightMargin: 20
}

}

```

تنظیم موقعیت توسط اتصالات (با اختصاص دادن عبارت‌های اتصال به محورهای x و y طول و ارتفاع به عنوان خواص یک شیء به جای استفاده از لنگرها) هرچند به حداقل انعطاف پذیری اجازه می‌دهد اما نتیجه تولیدی آن نسبتاً کُند است. در نظر داشته باشید که اگر شیء شما به صورت داینامیک (پویا) تغییر پیدا نمی‌کند، روش مناسب تغییر موقعیت استفاده از خاصیت‌های y، x، width و height می‌باشد که متناسب و وابسته به والد خود می‌باشد. در صورتی که می‌خواهید وابستگی ایجاد نشود بهتر است از لنگرها استفاده کنید.

در مثال زیر، اشیاء مستطیل (Rectangle) فرزند در یک مکان مشابهی قرار گرفته اند، اما کدهای مرتبط به لنگرها (anchors) نشان میدهد که موقعیت آن شیء به صورت ثابت مقدار دهی شده است.

```

import QtQuick 2.3

Rectangle {
    width: 60
    height: 60
    Rectangle {
        id: fixedPositioning
        x: 20
        y: 20
        width: 20
        height: 20
    }
    Rectangle {
        id: anchorPositioning
        anchors.fill: parent
        anchors.margins: 20
    }
}

```

```
}
```

## مدلها و نمایش‌ها

اکثر برنامه‌های کاربردی (Application) حداقل از یک مدل (Model) تغذیه داده را به نمایه (View) ارسال می‌کنند. بعضی از مواردی که توسعه دهنگان برای به حداقل رساندن عملکرد به آنها باید توجه داشته باشند وجود دارد.

## مُدل‌های سفارشی سمت سی‌پلاس‌پلاس

این بسیار خوب است که شما مدل‌های داده‌ای سفارشی خود را سمت C++ برای استفاده از نمایه QML بنویسید. اما باید توجه داشته باشید این کار به شدت بستگی به کاربرد آن در موقعیت خودش را دارد که برخی از دستورالعمل‌های کلی به شرح زیر هستند:

- تا جایی که ممکن است ناهمزمان باشد.
- تا جای ممکن بصورت ناهمزمان، تمام پروسه‌های پردازشی را در یک نخ با (اولویت پایین) انجام دهید.
- عملیات بکار راند را به صورت دسته‌ای انجام دهید تا (به طور بالقوه آهسته) عمل‌های I/O و IPC به حداقل برسد.
- از یک پنجره مجزا برای نتایج کش که پارامترهایش به پروفایلینگ کمک می‌کنند، استفاده کنید.

این مهم است که توجه داشته باشیم استفاده از یک نخ (Thread) کارآمد با کمترین اولویت توصیه می‌شود تا خطر قحطی (starving) را در نخ‌های رابط کاربری به حداقل برساند. همچنین به یاد داشته باشید که مکانیزم هماهنگ‌سازی و قفل کردن می‌تواند عامل مهمی برای عملکرد آهسته باشد، بنابراین لازم است از اعمال قفل غیر ضروری جلوگیری (صرف نظر) شود.

## نوع لیست مُدل (ListModel) در QML

کیوام‌آل (QML) یک نوع **ListModel** را فراهم می‌کند که می‌تواند برای ارسال داده به یک نوع از **ListView** استفاده شود. این باید برای اکثر موارد مناسب باشد و تا زمانی که به درستی استفاده شود نسبتاً عملکرد درستی داشته باشد.

### تجمع در داخل یک نخ

عناصر **ListModel** را می‌توان در یک نخ کاری (با اولویت پایین) در **JavaScript** جایجا کرد. توسعه‌دهنده باید صریحاً متده **sync()** را که به عنوان یکی از خواص موجود در **ListModel** می‌باشد را داخل یک **WorkerScript** صدا بزند تا تغییرات همزمان با نخ اصلی صورت بکیرد. جهت کسب اطلاعات بیشتر مستندات [WorkerScript](#) را مطالعه کنید.

لطفاً توجه داشته باشید که با استفاده از یک عنصر **WorkerScript** یک موتور جاوااسکریپت جداگانه ایجاد می‌شود (چرا که موتور جاوااسکریپت در هر نخ (Thread) می‌باشد. این باعث افزایش مصرف حافظه خواهد شد. چرا که عناصر متعدد (چندگانه) **WorkerScript** همه از یک نخ

استفاده خواهند کرد. بنابراین تاثیر شدیدی در حافظه استفاده شده توسط ۹ رکر دوم و سوم وارد خواهد کرد که در زمان اجرای برنامه و ۹ رکر اسکریپت اول تاثیر آن بسیار ناچیز خواهد بود.

## از نقش‌های (Roles) پویا استفاده نکنید

عنصر **ListModel** در ۲ **QtQuick** بسیار کارآمدتر از **ListModel** است. بهبود عملکرد عمدتاً از پیش فرض‌های مربوط به نوع نقشهای (**Roles**) در هر عنصر در یک مدل داده می‌شود. اگر نوع آن تغییر نکند، عملکرد ذخیره سازی به طور چشمگیری بهبود می‌یابد. اگر نوع قابلیت تغییر به صورت پویا از عنصر به عنصر دیگری را داشته باشد، بهینه سازی غیر ممکن شده و عملکرد (پرفمنس - کارآیی) مدل به اندازه بزرگی بدتر و گُندتر خواهد شد.

بنابراین به صورت پیشفرض حالت داینامیک غیر فعال بوده و توسعه‌دهنده خود باید خاصیت **dynamicRoles** را به عنوان یکی از خاصیت‌های **ListModel** فعال کند. البته به شدت پیشنهاد می‌شود برنامه خود را از اول طراحی کنید اما این قابلیت را فعال نکنید.

## نمایه‌ها (Views)

توجه داشته باشید که نماینده (دليگيتس يا delegates) باید به طور ساده حفظ شود. به اندازه کافی از انواع QML در اين خاصیت‌های ضروري استفاده کنيد. هرگونه قابلیت اضافه که بلافصله مورد استفاده قرار نمی‌گيرد نیاز نیست. برای مثال اگر نیاز است اطلاعات بیشتری در موقع کلیک بر روی آیتم نمایش داده شود، نیاز نیست که همان لحظه ایجاد و بر روی آیتم نمایان شوند. نباید تا قبل از زمان نیاز ایجاد شوند.

لیست زیر، خلاصه خوبی از مواردی است که باید هنگام طراحی بر روی خاصیت **delegate** در نظر داشته باشید:

- عناصری که کمتر در خاصیت **delegate** هستند، سریعتر می‌توانند ایجا شوند، و بنابراین سریعتر می‌توانند در نمایه (view) مشاهده و اسکرول شوند.
- تعداد پیوندها (اتصالات) را در **delegate** به حداقل تعدادشان نگه داری کنید. از لنگرهای (**anchors**) به جای اتصال برای موقعیت‌های نسبی در یک **delegate** استفاده شود.
- از به کار گیری عناصر **ShaderEffect** در **delegate** اجتناب شود.
- هرگز خاصیت **clip** را در خاصیت **delegate** فعال نکنید.

نکته: شما می‌توانید ویژگی **cacheBuffer** یک نمایه (view) را تنظیم کنید تا اجازه ایجاد و ارسال غیر مستقیم به خارج از ناحیه قابل مشاهده را ایجاد کنید. استفاده از **cacheBuffer** برای **delegate** های نمایشی توصیه می‌شود.

توجه داشته باشید که خاصیت **cacheBuffer** یک **delegate** اضافی را در حافظه نگه می‌دارد، بنابراین مقدار حاصل از استفاده **cacheBuffer** باید با استفاده از حافظه اضافی متعادل شود. توسعه‌دهندگان خود باید از معیارهای سنجش (بنچ مارک‌ها) برای یافتن بهترین مقدار مناسب برای

استفاده را انجام دهنده. چرا که افزایش حافظه ناشی از استفاده cacheBuffer می‌تواند در بعضی از موارد نادر، باعث کاهش نرخ فریم در هنگام پیمایش (اسکرول) شود.

## جلوه‌های بصری

فناوری **کیوت کوئیک ۲** شامل چند ویژگی است که به توسعه‌دهندگان و طراحان اجازه می‌دهد تا رابط کاربری فوق العاده جذاب ایجاد کنند؛ تغییرات پویا (داینامیکی) و همچنین جلوه‌های بصری می‌توانند برای یک اثر بزرگ در برنامه کاربردی مورد استفاده قرار گیرد. اما هنگام استفاده از بعضی از ویژگی‌های **QML**، باید نکاتی را در نظر گرفت که می‌توانند دلالت بر کارآیی (پرفرمنس) را داشته باشند.

## انیمیشن‌ها

به طور کلی، متحرک سازی (انیمیشن سازی) یک خاصیت (پرایپری) می‌تواند سبب شود تا اتصالاتی که به یک خاصیت دیگر اشاره می‌کند را مجدداً مورد ارزیابی قرار گیرد. معمولاً این مورد مطلوب است، اما در برخی از موارد ممکن است بهتر باشد قبل از انجام متحرک سازی (انیمیشن سازی) اتصالات را غیر فعال کنید، و سپس آن انیمیشن را تکمیل کنید.

از اجرای کدهای جاوااسکریپت در طول یک انیمیشن اجتناب کنید. برای مثال، اجرای یک عبارت پیچیده جاوااسکریپت برای هر فریم از انیمیشن یک خاصیت مانند  $\times$  باید اجتناب شود. توسعه‌دهندگان باید در استفاده از انیمیشن‌های اسکریپتی دقیق باشند، زیرا آنها در یک نخ اصلی اجرا می‌شوند (در نتیجه در صورتی که اجرا و تکمیل انیمیشن بیش از حد طول بکشد)، ممکن است فریم‌هایی را پرش کرده و موجب کاهش کارآیی اصلی آن شود.

## ذرات، پارتیکل (Particles)

ماژول **Qt Quick Particles** اجازه می‌دهد تا ذراتی را برای زیبایی هرچه بهتر رابط کاربری اضافه شود. با این حال، هر پلتفرم دارای قابلیت‌های سخت افزاری مختلفی است و ماژول **Particles** قادر به محدود کردن پارامترهای سخت افزاری شما نمی‌باشد. بنابراین زمانی که شما ذرات بیشتری برای تولید (رندر) می‌خواهید (هرچه بزرگتر می‌شوند)، سرعت بیشتری برای پردازش سخت افزار گرافیکی شما نیاز خواهد بود تا بتوانند سرعت تولید آنها را با نرخ ۶۰ فریم بر ثانیه اجرا کنند. بنابراین ذرات بیشتر خواهان پردازنده‌های مرکزی سریعتری می‌باشند.

بنابراین، این بسیار مهم است که تمام ذرات و اثرات آنها را بر روی پلتفرم‌های هدف خود با دقیق آزمایش کنید، تا برای کالیبره کردن تعداد و اندازه ذرات قابل تولید با نرخ ۶۰ فریم به نتیجه قابل قبول برسید.

لازم به ذکر است که برای جلوگیری از شبیه‌سازی غیر ضروری یک سیستم ذره را می‌توان غیر فعال کرد (به عنوان مثال عنصری را غیرقابل مشاهده کنید) این کار باعث می‌شود تا شبیه‌سازی‌های غیر ضروری غیر فعال شوند.

عملکرد سیستم ذرات با مقایس تعداد ذرات نگه داری می‌شود. پس از نمونه برداری از اثر مورد نظر، با کاهش مقدار ذرات، عملکرد آن را می‌توان بهبود داد. بر عکس، اگر عملکرد آن به خوبی و در حد قابل قبول باشد، می‌توان تعداد ذرات را تا زمانی که در آن نقطه قرار می‌گیرد افزایش دهید (این کار باعث بهبود خواهد شد).

همانند **ShaderEffect**، عملکرد سیستم ذرات تا حد زیادی وابسته به ساخت افزار گرافیکی است که در حال اجرا می‌باشد.

## کنترل طول عمر عنصر

با تقسیم کردن یک برنامه به اجزای ساده، مازولار، هر کدام از آنها در یک فایل **QML** منعکس می‌شوند، می‌توانید زمان راه اندازی برنامه را سریعتر و کنترل بیشتری از استفاده از حافظه را به دست آورده و تعداد عناصر فعل اما غیر قابل مشاهده را در برنامه خود کاهش دهید. به طور کلی سعی کنید برای هر مازول یا بخشی از برنامه یک سند جداگانه از QML را فراهم کنید.

### مقداردهی اولیه از روی تنبلی!

موتور QML برخی چیزها را به صورت هوشمندانه (یا زیرکانه) انجام می‌دهد تا اطمینان حاصل شود که بارگذاری و مقدار دهی اولیه اجزاء باعث نمی‌شود تا فریم‌ها از بین بروند. با این حال هیچ راهی برای کاهش زمان راه اندازی بهتر از این وجود ندارد که شما کارهایی را تا زمانی که به آن‌ها نیاز ندارید را انجام ندهید و از آن‌ها اجتناب کنید. این ممکن است با استفاده از اجزای پرکاربردی مانند نوع **Loader** یا ساخت کامپوننت (جزء) های پویا (**Dynamic Object Creation**) انجام شود.

### استفاده از بارگذار (Loader)

لودر یک عنصری است که اجازه می‌دهد بارگذاری و تخلیه پویا بر روی اجزاء انجام شود.

- با استفاده از ویژگی "active" در **Loader**، مقدار دهی اولیه می‌تواند تا زمان لازم انجام شود.
- با استفاده از تابع `setSource()` مقدار دهی اولیه می‌تواند مشخص شود.
- تنظیم خاصیت `asynchronous` بر روی مقدار `true` می‌تواند تاثیر بر روی بهبود عملکرد بر روی کامپوننت (جزء) در زمان نمونه سازی داشته باشد.

### استفاده از سازنده پویا

توسعه دهنگان می‌توانند از یک تابع (`Qt.createComponent()` برای ایجاد یک مولفه به صورت پویا در زمان اجرا از داخل جاوااسکریپت استفاده کنند و سپس (`createObject()` را برای نمونه سازی آن، فراخوانی کنند.

نایبود کردن عناصر استفاده نشده

عناصری که مخفی هستند به عنوان فرزندی از عناصر غیر بصری محسوب می‌شوند. برای مثال زمانی که یک زبانه از **TabWidget** یا **TabBar** انتخاب شده است و نمایش داده می‌شود به دلایلی باید سریع مقدار دهی اولیه شوند و زمانی که از آن به مدت طولانی استفاده نمی‌شود و این خود باعث بارگذاری اضافی است. بنابراین در صورت استفاده از این روش جهت جلوگیری از این هزینه مدام در زمان اجرا (که شامل تولید انیمیشن، اتصالات، رندرینگ و غیره...) به صورت خودکار حذف می‌شوند.

نکته: یک آیتم بارگذاری شده با یک نوع عنصر **Loader** ممکن است توسط تنظیمات مجدد خاصیت `sourceComponent` یا `source` آزاد شود. در حالی که موارد دیگر ممکن است به صورت صریح با فراخوانی متدهای `destroy()` بر روی آنها منتشر شود. در بعضی از موارد ممکن است لازم باشد آیتم فعال را ترک کرده و یا حداقل آن را نامرئی کنید. این کار موجب می‌شود سرعت برنامه شما بهینه شود.

## تولید (Rendring)

گرافیک صحنه‌ای که برای رندر در کیوت کوئیک ۲ استفاده می‌شود، رابط کاربری بسیار متحرک را به صورت فیزیکی در ۶۰ فریم بر ثانیه ارائه می‌کند. با این حال برخی چیزها به طور چشمگیری در عملکرد رندرینگ کاهش می‌بابند. توسعه دهندگان باید مراقب باشند تا از این مشکلات در هر جا که ممکن است اجتناب کنند که به برخی از آن‌ها اشاره شده است.

## کلیپ کردن (Clipping)

کلیپ کردن به صورت پیشفرض غیرفعال است و توصیه می‌شود تنها زمانی که به آن نیاز دارید فعالش کنید.

کلیپ کردن یک اثر بصری دارد، نه بهینه سازی! بنابراین این ویژگی پیچیدگی بیشتری را برای رندرینگ افزایش می‌دهد. اگر کلیپ فعال باشد، اجزای تولید شده خود و دیگر اجزای فرزندش را به محدوده خود متصل خواهد کرد. این باعث می‌شود رندرینگ در آن بخش متوقف شده و آن بخش از اشیاء مرتبط و منظم دیده شوند.

نکته مهم این است که کلیپ کردن در داخل `delegate` بسیار نامناسب است و باید از این کار اجتناب شود. چرا که موجب کاهش کارآیی برنامه خواهد شد.

## نقاشی بیش از اندازه و عناصر نامرئی

اگر شما عناصری دارید که توسط عناصر دیگری (مات) یا پوشانیده شده اند، بهتر است از خاصیت `visible` استفاده کرده و آن را به مقدار `false` تنظیم کنید. برای مثال در زبانه‌هایی که به صورت پیشفرض یکی از آن‌ها فعال هستند و زبانه‌های دیگر تا زمان انتخاب مخفی! در این صورت بهتر است آیتم‌های آن‌ها به از طرف والد آن‌ها مخفی شوند تا در زمان اجرا بابت نمایش مواردی که مخفی هستند هزینه‌ای نشود.

## شفاف در مقابل مات

محتوای مبهم (مات) عموماً بسیار سریعتر از محتوای شفاف هستند. دلیل این امر این است که محتوای شفاف نیاز به ترکیب (مخلوط) کردن دارد و سیستم رندر کننده به مراتب می‌تواند نوع مات را بهتر بهینه سازی نماید. یک تصویر با یک پیکسل شفاف به طور کامل به عنوان یک نتیجه کاملاً شفاف تلقی می‌شود، حتی اگر عمدتاً مات باشد. همین امر برای نوع **BorderImage** بالهای شفاف درست است.

## سایه‌ها

نوع **ShaderEffect** باعث می‌شود که کد درون خطی **GLSL** را به صورت یکپارچه در یک برنامه **Qt Quick** با سریار بسیار کم قرار دهید. با این حال مهم است که بدانید، که قطعه برنامه نیاز به اجرا برای هر یک از پیکسل‌ها در شکل تولید شده را دارد.

هنگام استقرار بر روی یک سخت افزار با توان پایین، شیدرها مقدار زیادی از پیکسل‌ها را پوشش می‌دهند برای جلوگیری از عملکرد ضعیف، باید یک قطعه شیدر را برای چند دستورالعمل جهت جلوگیری از پرفرمنس ضعیف نگهداری کنید. شیدرهای نوشته شده در `GSL` اجازه می‌دهد تا تبدیلات و جلوه‌های پیچیده‌تری را تولید کنید. با این حال باید با دقت بسیاری از آنها استفاده کرد.

استفاده از `ShaderEffectSource` باعث از پیش رندر شدن صحنه به `FBO` قبل از کشیده شدن آن می‌شود. این سربار اضافی می‌تواند بسیار پرهزینه باشد.

## تخصیص و جمع آوری حافظه

مقدار حافظه‌ای که توسط یک برنامه اختصاص داده می‌شود و اینکه آن حافظه چگونه اختصاص داده می‌شود شامل ملاحظات بسیار مهمی هستند. صرف نظر از نگرانی‌های مربوط به خارج از حافظه در دستگاه‌های محدود به حافظه، تخصیص حافظه در پُشته به عنوان یک عمل محاسباتی نسبتاً گران می‌باشد. در این میان استراتژی‌های اختصاصی تخصیص حافظه می‌تواند باعث افزایش تقسیم داده‌ها در صفحات شود. خوشبختانه، جواالاسکریپت از روش مدیریت حافظه خودکار در پُشته (Heap) در قالب `GC` پشتیبانی می‌کند که دارای برخی از مزیت‌ها است. اما با این حال دلالت بر مهم بودن برخی موارد دارد.

یک برنامه نوشته شده در QML حافظه را از هر دو پُشته از سمت مدیریت حافظه تحت `C++` و `JavaScript` مدیریت می‌کند. توسعه دهنده نرم‌افزار باید در رابطه با هر یک از آنها به منظور به حداقل رساند پرفرمنس آگاه باشد.

## نکاتی برای توسعه‌دهندگان برنامه QML

نکات و پیشنهادات موجود در این بخش تنها در قالب دستورالعمل هستند و ممکن است در همه شرایط قابل اجرا نباشند. با استفاده از معیارهای تجربی و بررسی بنج مارک‌ها و همچنین آنالیز و همچنین با استفاده از ومعیارهای تجربی برنامه خودتان بهترین تصمیم ممکن را بگیرید.

اگر برنامه شما شامل نمایه (View) های متعدد (به عنوان مثال زبانه‌های چندگانه) می‌باشد اما در هر زمان تنها یکی از آنها مورد نیاز است در این صورت برای کم کردن حافظه مصرفی می‌توانید از روش مقداردهی از روی تبلی استفاده کنید که در بالا به آن اشاره شده است.

## نابود سازی اشیاء ای که مورد استفاده قرار نمی‌گیرند

اگر شما از روش ساده بارگیری کامپوننت‌ها و یا ساخت اشیاء به صورت پویا در طول یک عبارت جواالاسکریپتی استفاده می‌کنید، بهتر است آنها را به صورت دستی توسط متدهای `destroy()` نابود کنید. این بهتر از آن است که منتظر باشید تا به صورت خودکار سیستم `GC` آنها را جمع آوری و نابود کند.

## در زمانی که نیاز نباشد به صورت دستی عمل (GC بازیافت حافظه) را انجام دهید

در اکثر موارد، دستیابی به مجموعه زباله‌ها به منظور دستیابی به آن نیست. زیرا این کار به مدت زیادی نخهای سمت رابط کاربری را مسدود می‌کند. این کار باعث می‌شود فریم‌ها و پرش‌هایی در انیمیشن‌های متحرک به وجود آید که باید از اینگونه هزینه‌ها اجتناب شود. به طور کلی باید توجه داشته باشید که به صورت مستقیم و دستی همه جا نباید اقدام به نابود سازی حافظه اخلاص یافته شده کنید.

## اجتناب از اتصال پیچیده

گذشته از اینکه اتصالات پیچیده موجب کاهش پرفرمنس (کارایی) می‌شود استفاده از آن‌ها (برای مثال، زمانی که نیاز است ارزیابی جداگانه تحت کدهای جاوااسکریپت شود (به صورت پیچیده‌ای حافظه بیشتری را در هر دو پُشته C++ و JavaScript برای بهینه سازی محاسبات اختصاص می‌دهند. بنابراین نیاز نیست همیشه از اتصالات پیچیده و ارزیابی آن‌ها تحت JS استفاده کرد.

## اجتناب از تعریف چند نوع ضمنی یکسان

اگر یک عنصر QML یک خصوصیت سفارشی تعریف شده در QML داشته باشد، آن نوع خاصی و ضمنی خود را می‌گیرد. این مورد در بخش بعدی بیشتر توضیح داده شده است. اگر چندین نوع ضمنی یکسان در یک جزء درون خطی تعریف شده باشند، موجب سرآیندرفتن بخشی از حافظه خواهند شد. در این وضعیت معمولاً بهتر است که جزء را به صورت صریح تعریف کنیم که بعداً می‌تواند دوباره استفاده شود.

تعریف یک خاصیت سفارشی اغلب می‌تواند در بهینه سازی عملکرد سودمندی داشته باشد (برای مثال، برای کاهش تعداد پیوندهایی که مورد نیاز است یا دوباره ارزیابی می‌شوند)، یا می‌تواند مازولار بودن و قابلیت نگهداری یک جزء را بهبود بخشد. در این موارد، استفاده از خواص سفارشی پیشنهاد می‌شود. با این حال، نوع جدید باید، اگر از بیش از یک بار استفاده می‌شود، به جزء خود (فایل.qml) به منظور حفظ حافظه، تقسیم شود.

## استفاده مجدد از اجزای موجود

اگر شما در حال تعریف یک جزء جدید هستید، لازم است دوباره بررسی کنید که چنین جزئی در پلتفرم شما وجود دارد یا خیر. در غیر این صورت شما باید موتور QML را مجبور به ساخت و ذخیره سازی نوع داده برای یک نوع که نیاز است کنید که به عنوان یک نوع تکراری که از اجزای موجود می‌باشد بارگذاری می‌شود.

- به جای کتابخانه‌های اسکریپتی پرآگما از انواع تک تک (singleton) استفاده کنید.
- اگر از یک اسکریپت کتابخانه pragma برلی ذخیره داده‌های نمونه استفاده می‌کنید، به جای استفاده از یک نوع تک تکی از **QObject** استفاده کنید. نتیجه آن در کارایی بهتر تاثیر گذار خواهد بود و باعث می‌شود حافظه کوچکتری در پُشته جاوااسکریپت مورد استفاده قرار گیرد.

## اختصاص دادن حافظه در یک برنامه QML

استفاده از حافظه یک برنامه مبتنی بر **QML** ممکن است به دو بخش تقسیم شود: استفاده از پُشته سمت **C++** و پُشته سمت **JavaScript** می‌باشد. برخی از حافظه اختصاص داده شده در هر یک از اجزاء غیرقابل اجتناب خواهند بود. به عنوان آن که توسط موتور QML یا موتور جاواسکریپت اختصاص داده می‌شود. در حالی که بقیه آن وابسته به تصمیمات گرفته شده توسط توسعه دهنده اپلیکیشن می‌باشد.

پُشته در C++ شامل موارد زیر خواهد بود:

- سربار ثابت و اجتناب ناپذیر از موتور QML (پیاده سازی ساختارهای داده، اطلاعات زمینه و غیره.)
- اطلاعات هر جزء کامپایل شده و نوع داده‌ها، شامل هر یک از انواع و فرا داده‌های می‌باشد، که توسط موتور QML بسته به نوع مازولها و کامپوننت‌هایی است که توسط اپلیکیشن بارگیری می‌شوند.
- هر شیء‌ای که در داده‌های سی<sup>++</sup> (شامل مقادیر خاصیت) به همراه هر یک از عناصر متا آبجکت هستند که وابسته به کامپوننت (اجزای معرفی شده توسط اپلیکیشن می‌باشند).
- هر داده‌ای که به طور خاص توسط QML اختصاص داده می‌شود که شامل کتابخانه‌های وارد شده نیز هستند.

پُشته در JavaScript شامل موارد زیر خواهد بود:

- سربار ثابت و اجتناب ناپذیر از موتور QML (خودش انواع از قبل ساخته شده جاواسکریپتی را دارد.)
- سر بار ثابت شده و اجتناب ناپذیر از ادغام جاواسکریپت (توابع سازنده برای انواع داده‌های بارگذاری شده، قالب‌های توابع، و غیره.)
- اطلاعات مربوط به هر نوع و دیگر انواع داخلی توسط موتور جاواسکریپت که در زمان اجرا تولید می‌شود.
- هر شیء‌ای که به عنوان داده جاواسکریپتی (خاصیت‌های نوع var ، توابع جاواسکریپتی و هندرلهای سیگنال و عبارات بهینه نشده).
- متغیرهای اختصاص داده شده در زمان ارزیابی عبارت

علاوه بر این، یک پُشته جاواسکریپتی اختصاص یافته شده برای استفاده از نَخ اصلی و دیگری در پُشته جاواسکریپت برای استفاده در اختصاص یافته می‌شود. اگر یک اپلیکیشن از یک عنصر **WorkerScript** استفاده نکند متحمل به سربار گیری نخواهد شد. اندازه پُشته در جاواسکریپت می‌تواند به چندین مگابایت برسد و بنابراین برنامه‌های نوشته شده برای دستگاه‌هایی که دارای محدودی حافظه هستند، ممکن است بهترین با اجتناب از عنصر **WorkerScript** باشد، با وجود سودمندی آن در مدل‌ها لیستی، که به صورت یکپارچه ذخیره می‌شوند باشد.

توجه داشته باشید که هر دو موتور **QML** و **JavaScript** به طور خوکار مخازن خود را از نوع داده‌های ملاحظه شده تولید خواهند کرد. هر کامپوننت (جزء) توسط یک برنامه بارگذاری می‌شود که یک نوع متمایز (صریح) بوده و هر عنصر (به جای کامپوننت) ویژگی‌های سفارشی خود را در QML که به صورت ضمنی است تعریف می‌کند.

مثال زیر را در نظر بگیرید:

```

import QtQuick 2.3

Item {
    id: root

    Rectangle {
        id: r0
        color: "red"
    }

    Rectangle {
        id: r1
        color: "blue"
        width: 50
    }

    Rectangle {
        id: r2
        property int customProperty: 5
    }

    Rectangle {
        id: r3
        property string customProperty: "hello"
    }

    Rectangle {
        id: r4
        property string customProperty: "hello"
    }
}

```

نمونه‌های قبلی مستطیل‌های r0 و r1 دارای ویژگی‌های اختصاصی (سفارشی) نبودند. بنابراین موتورهای جاوااسکریپت و QML هر دو آنها را از همان نوع در نظر می‌گیرند. به عبارت دیگر هر دوی آن‌ها به عنوان یک نوع صریح از **Rectangle** مستطیل در نظر گرفته می‌شوند. مستطیل‌های r2، r3 و r4 هر یک از آنها با داشتن ویژگی‌های اختصاصی خود هر کدام با انواع مختلف ضممنی در نظر گرفته شده‌اند. حتی اگر اطلاعات مربوط به ویژگی‌های یکسانی داشته باشند.

## ملاحظات اختصاصی در ُعمق حافظه

هرگاه تصمیماتی در خصوص تخصیص دادن حافظه یا کارآیی آن به وجود آید، این مهم است که تاثیرات شدید در عملکرد پردازنده مرکزی و مخازن مربوط به آن، صفحه بندی‌های سیستم‌عامل و بازیافت حافظه (GC) در جاوااسکریپت را در نظر داشته باشد. بنابراین راه حل‌های بالقوه باید با دقت بررسی شوند تا مطمئن شوید که بهترین مورد را انتخاب کرده‌اید.

هیچ مجموعه‌ای از دستور العمل‌های کلی نمی‌تواند یک درک جامع ای از اصول اساسی علوم رایانه را با یک دانش علمی از جزئیات پیاده سازی کرده و پلتفرمی که یک توسعه‌دهنده نرم‌افزار در حال توسعه آن است را جایگزین کند.

## تقسیم بندی

تقسیم بندی به عنوان یک مسئله برای توسعه در سی‌پلاس‌پلاس است. اگر توسعه‌دهنده برنامه هیچ نوع یا پلاگینی از سی‌پلاس‌پلاس را تعریف نکند، ممکن است آنها را در این بخش با خیال راحت نادیده بگیرند.

با گذشت زمان، یک برنامه بخش بزرگی از حافظه را برای خود اختصاص داده و داده‌ها را به آن حافظه ارسال می‌کند و بعضی اوقات برخی از قسمت های آن را پس از اتمام استفاده آن‌ها را آزاد می‌کند. این می‌تواند منجر به «حافظه آزاد» شده‌ای در تکه‌های غیر مجاور شود که نمی‌تواند برای برنامه‌های کاربردی دیگر توسط سیستم‌عامل مورد استفاده قرار گیرد. همچنین تاثیر شدیدی بر روی ذخیره سازی پنهان و دسترسی به ویژگی‌های یک اپلیکیشن می‌گذارد. زیرا داده‌هایی که زنده هستند (در حال استفاده) هستند، ممکن است در بسیاری از صفحات مختلف حافظه فیزیکی گسترش یابند. این به نوبه خود می‌تواند سیستم‌عامل را مجبور به (swap) یا مبادله کند که می‌تواند سبب شود تا عمل **O/I** ایجاد شود که به شدت عمل آهسته‌ای بشمار می‌آید.

تقسیم‌بندی می‌تواند توسط دیگر موارد تخصیص دهنده حافظه، با کاهش میزان حافظه‌ای که در هر زمان با دقت مدیریت زمان زندگی اشیاء مورد بررسی قرار گیرد. با تمیز کردن و باز سازی دوره‌ای از حافظه‌ها یا با استفاده از زمان اجرا بر روی حافظه از تجزیه آن می‌توان جلوگیری کرد که توسط سیستم‌عامل بازیافت حافظه خودکار (GC) مانند جاواسکریپت ممکن است.

## بازیافت حافظه

جاواسکریپت سیستم بازیافت حافظه به صورت خودکار را فراهم می‌کند. حافظه‌ای که در دسته جاواسکریپتی قرار دارد (برخلاف پُشته در سی‌پلاس‌پلاس (متعلق به موتور خود جاواسکریپت است. این موتور به طور دوره‌ای تمامی داده‌های ناشخص (غیر قابل استفاده) را در پُشته جاواسکریپت جمع آوری می‌کند.

## پیامدهای بازیافت حافظه

بازیافت حافظه، مزایا و معایبی دارد. این به این معنی است که مدیریت عمر مفید شیء به صورت دستی اهمیت کمتری دارد. با این حال، این بدان معنی است که یک عمل بالقوه طولانی مدت ممکن است توسط موتور جاواسکریپت در زمانیکه که خارج از کنترل توسعه‌دهنده نرم‌افزار است آغاز شود. استفاده از پُشته در جاواسکریپت با دقت بسیاری توسط توسعه‌دهنده برنامه مورد توجه قرار می‌گیرد. لذا تکرار و مدت زمان بازیابی حافظه ممکن است تاثیر منفی بر تجربه نرم‌افزار داشته باشد.

## فراخوانی بازیابی حافظه

یک برنامه نوشته شده در QML (به احتمال زیاد) در یک مرحله‌ای به بازیافت حافظه (GC) نیاز دارد. در صورتی که حجم حافظه آزاد شده کم باشد سیستم بازیافت خودکار حافظه توسط موتور جاواسکریپت انجام می‌شود. بعضی اوقات این کار در صورتی که توسعه‌دهنده نرم‌افزار تصمیمی در مورد اینکه چه زمانی بازیافت حافظه را انجام دهد نگرفته باشد می‌تواند مناسب باشد. (اگر چه که معمولاً این مورد مطرح نمی‌شود).

توسعه‌دهنده نرم‌افزار احتمالاً می‌داند که برنامه چه زمانی را به مدت قابل ملاحظه‌ای بیکار است. اگر یک برنامه QML از حافظه بسیار زیادی از پُشته جاواسکریپت را مصرف کند، باعث می‌شود چرخه و تکرارهای مکرری در بازیافت حافظه صورت گیرد که در وظایف حساس بر روی کارآیی تاثیر خواهد گذاشت. مانند (لیست پیمایش، انیمیشن‌ها، وغیره). توسعه‌دهنده نرم‌افزار ممکن است به طور دستی به بازیافت حافظه در طول دوره صفر فعالیت کم کند. دوره‌های بیکاری برای انجام بازیافت حافظه ایده آل هستند چراکه کاربر هیچ‌گونه تضعیفی را در تجربه کاربری خود (فریم‌های پرش شده، انیمیشن‌های پرتحرک و نامنظم وغیره) حس نخواهد کرد؛ که این تضعیف تجربه، درنتیجه فراخوانی بازیافت حافظه به هنگام فعالیت برنامه رخ می‌هد.

بازیافت حافظه ممکن است به صورت دستی با فراخوانی (gc) در جاوا اسکریپت اعمال شود. این باعث می‌شود یک چرخه بازیافت جامع از حافظه صورت بگیرد که ممکن است مدت زمانی بین چند صد تا بیش از هزار میلی ثانیه برای تکمیل آن سپری شود. بنابراین در صورت امکان باید از آن اجتناب شود.

## حافظه در مقابل کارآیی

در بعضی موارد ممکن است که زمان پردازش حافظه برای سبک سنگین کردن افزایش مصرف حافظه کاهش باید. برای مثال، ذخیره سازی نتیجه یک جستجوی نماد که در یک حلقه تنگ به یک متغیر موقت در یک عبارت جاوا اسکریپتی استفاده می‌شود که در بهبود ارزیابی این عبارت تاثیر قابل توجهی خواهد داشت. اما این شامل تخصیص یک متغیر موقت می‌باشد. در بعضی از موارد، این سبک سنگین کردن ممکن است معقول باشد (مانند موارد فوق، که تقریباً همیشه منطقی هستند)، اما در موارد دیگر ممکن است بهتر باشد تا اجازه دهیم تا پردازش طول بکشد تا از افزایش فشار بر روی حافظه سیستم جلوگیری شود.

در بعضی از موارد، تاثیر افزایش فشار بر روی حافظه می‌تواند شدید باشد. در برخی موارد، استفاده دوباره از حافظه ممکن است برای به دست آوردن عملکرد پیشنهادی منجر به افزایش ترافیک صفحات یا ترافیک بر روی حافظه نهان شود که باعث کاهش عملکرد شدید آن خواهد شد. این همیشه برای ارزیابی لازم است، تاثیر تعاملات با دقت به منظور تعیین اینکه بهترین راه حل در یک وضعیت خاص است مهم هستند.

## قانون حمایت حقوق مولفان و مصنفان و هنرمندان

ماده ۱- از نظر این قانون به مولف و مصنف و هنرمند «پدید آورنده» و به آنچه از راه دانش یا هنر و یا ابتکار آنان پدید می‌آید بدون در نظر گرفتن طریقه یا روشهای در بیان و یا ظهور و یا ایجاد آن به کار رفته «اثر» اطلاق می‌شود.

ماده ۲ - حقوق پدید آورنده شامل حق انحصاری نشر، پخش، عرضه و اجرای اثر و حقوق بهره‌برداری مادی و معنوی از نام و اثر است.

بنابراین هرگونه کپی برداری از این کتاب و تکثیر آن بدون اطلاع نویسنده و شرکت توسعه‌دهنده پیگرد قانونی خواهد داشت، لذا خواهشمند است در صورت مشاهده و یا دریافت نسخه‌هایی از این کتاب در منابع غیر مجاز آن را به ما اطلاع دهید. چرا که تنها منابع رسمی برای انتشار این کتاب مرجح رسمی نویسنده و ناشر آن است.

<https://kambizasadzadeh.com> | <https://genyleap.com> | <https://iostream.ir>

kambiz.ceo@gmail.com | @KambizAsadzadeh

## برخی از منابع و مراجع علمی جهت توسعه کتاب

<http://iso.cpp.org>

<https://gcc.gnu.org>

<http://en.cppreference.com>

<http://cplusplus.com>

<http://docs.qt.io>

<http://qt.io/qt-quick>

<http://qt.io/qtqml-index.html>

<http://qmlbook.org>

<http://json.com>

<http://wikipedia.com>

<http://developer.apple.com>

<http://www.kdab.com>

<http://microsoft.com>

<http://www.w3.org/TR/SVG>

<https://javascript.info>

<https://developer.ubuntu.com>

<http://developers.google.com>

<http://www.w3schools.com>

<https://iostream.ir>

# ممنونم!

خیلی ممنون از شما که این کتاب رو خوندین، امیدوارم که این کتاب به دردون خورده باشه ☺ برای دیدن بقیه مقالات و اطلاعات در رابطه با برنامه‌نویسی مدرن می‌توانید وارد وبسایت [جامعه برنامه‌نویسان مدرن \(iostream.ir\)](#) بشید. اگه به مشکلی برخوردین که من می‌تونم کمکتون کنم یا اگه متوجه شدین که در گفته‌های من اشتباهی وجود داره، در تؤییتر یا دیگر شبکه‌های اجتماعی بهم خبر بدین. نام کاربری من تو تؤییتر [@KambizAsadzadeh](https://KambizAsadzadeh) هست. در ضمن می‌توانید از پُل‌های ارتباطی دیگه‌ای هم استفاده کنید که در سایت [جامعه ما ذکر شده](#).

من در رابطه با اهداف و محصولاتی که بتونه در [بومی‌سازی](#) و [تولید سریع](#) و [توسعه استارت‌آپ‌ها](#) کمک کنه در [و بلاگ جامعه برنامه‌نویسان ایران](#) به آدرس [iostream.ir](#) منتشر می‌کنم.

برای اینکه یک برنامه‌نویس خوب و مفید باشید آستین‌هاتون رو بالا بزنید و از همین الان دست به کار بشین. تولید و توسعه محصولات خوب که بتونه در حد استانداردهای جهانی باشه نیاز به دانش و رعایت اصول رو داره که همراه با تجربه و دست به کار شدن به دست می‌اد. نگران این نباشید که مهندس کامپیوتر هستید یا نه، مساله اصلی اینه که علم برای همست و ما باید هر چیزی رو که یاد می‌گیریم بخشی از اون رو در اختیار بقیه قرار بدیم تا بتونیم دنیای خودمون رو بسازیم.

می‌دونم سوال‌های خیلی زیادی وجود داره و شاید انتظار داشته باشید که مطالب بیشتر و جامع‌تری در اختیار داشته باشید، اما من به عنوان نویسنده این کتاب به یه چیزی اعتراف می‌کنم که فقط با کتاب خوندن نمی‌توانیم موفق بشیم مهم عمل و تجربست. من این کتاب رو نوشتم تا حداقل سر نخی رو از نحوه برنامه‌نویسی با فناوری‌های پیشرفته در اختیارتون قرار بدم تا شما ادامه مسیر رو پیدا کرده و خیلی بهتر از این تجربه کنید. پس نگران نباشید و تنها با تکیه به خداوند متعال و باور اینکه شما می‌توانید شروع کنید هیچوقت هم پا پس نکشید چون هر وقت دیدین حوصلتون سر رفته یا دیگه خسته شدین مطمئن باشید همون لحظه است که چند قدم با موفقیت فاصله دارید.

شاد باشید

کامبیز اسدزاده

## سی‌پلاس‌پلاس (C++) با آوای (سی‌پلاس‌پلاس) یک زبان مدرن و قدرتمند غالب ساخت بشر تا به امروز است که هدف نوشتن

برنامه کارا و ساخت یافته و همچنین موضوعی یا (شیء گرا) را برای برنامه نویسان فراهم می‌کند. این نام منسوب به ریک ماسکیتی (اواسط ۱۹۸۳) است و برای اولین بار در دسامبر سال ۱۹۸۳ به کار برده شد. در طول مدت تحقیق این زبان بنام «C جدید» و بعدها «با کلاس» خوانده شد. در علوم کامپیوتر هنوز هم C++ به عنوان ابرساختار C شناخته می‌شود. آخرین نام از عملگر ++ در زبان C که برای افزایش مقدار متغیر به اندازه یک واحد بکار می‌رود و یک عرف معمول برای نشان دادن افزایش قابلیت‌ها توسط + ناشی گشته است. با توجه به نقل قولی از استراتس‌تروپ سازنده سی‌پلاس‌پلاس : «این نام ویژگی‌ها تکاملی زبان در C را نشان می‌دهد.» C+ نام زبانی غیرمرتبط به این زبان است.

## کیوت (Qt) مجموعه‌ای از کتابخانه‌ها و سرآیندهای نوشته شده به زبان C++ است که به برنامه‌نویس امکان توسعه آسان

نرم‌افزارهای کاربردی را می‌دهد. بنابراین توجه داشته باشید کیوت یک زبان نیست ! لذا در تمامی مراحل تولید یک محصول شما به زبان C++ در حال کد نویسی خواهد بود همچنین کیوت شامل چندین کلاس برای کار با واسط گرافیکی، چندسانه، ابزارهای پایگاهداده، شبکه و ... است.

نرم‌افزارهای نوشته شده با ابزار کیوت قادرند تا با استفاده از یک کامپایلر زبان C++ برای طیف وسیعی از سیستم‌عامل‌ها از جمله گنو/لینوکس (نسخه‌های رومیزی و وسیله‌های قابل حمل)، ویندوز، ویندوز CE، مک‌اواس و ... همگرданی شوند. بدین ترتیب حمل نرم‌افزار نوشته شده بدون تغییر در متن کد نوشته شده امکان‌پذیر است که همانند کتابخانه‌های دیگر Boost، STL، C++ مانند Poco، wxWidgets و ... مورد استفاده قرار می‌گیرد با تفاوت اینکه در زمینه طراحی رابط گرافیکی نسبت به کتابخانه‌های دیگر قدرتمند عمل می‌کند.

کامبیز اسدزاده، برنامه‌نویس و کارآفرین، مشاور و منتور فنی در حوزه مهندسی کامپیوتر و فناوری‌های مرتبط با آن است. او موسس شرکت جنی‌لیپ بوده و طی سال‌ها تجربه‌ها را جهت زمینه‌سازی برای راه اندازی کسب‌وکارهای بومی فراهم ساخته است. از محصولات برجسته او می‌توان به موتور سِل تحت زبان سی‌پلاس‌پلاس اشاره کرد که به نوعی یک شتابدهنده نرم‌افزاری در محصولاتش جهت تولید و توسعه محصولات نرم‌افزاری در تمامی پلتفرم‌ها در کمترین زمان ممکن است. همچنین می‌توان به سیستم فانوکس اولین پلتفرم آموزشی چند منظوره، سیستم تیگرای او اشاره کرد که یک سیستم یکپارچه نرم‌افزاری در حال توسعه هستند.

از نظر او برای توسعه و موفقیت جامعه استارت‌اپی باید تعامل و ارزش بین تخصص‌های فنی و کسب‌وکار حفظ شود تا تیم‌های استارت‌اپی بتوانند در ساده‌ترین حالت ممکن با بالاترین تعبیه میزان رشد و تولید استارت‌اپ‌های مفید را افزایش دهند. از این رو این کتاب برنامه‌نویسی پیشرفته یکی از پیشنهاداتی است که او برای علاقه‌مندان حوزه برنامه‌نویسی هدفمند و پیشرفته تحت فناوری چند-سکویی پیشنهاد می‌کند.

