Qt Fundamentals: Debugging

Printing to the console

To print, use the **qDebug** function.

- It is always available, but can be silenced when build for release
- Works like **printf** or **std::cout** but automatically appends '\n'
- Using **qPrintable** macro to print **QString**'s

```
qDebug("Integer value: %d", 42);
qDebug("String value: %s", qPrintable(myQString));
```

- Can be used with streaming operators:

```
#include <QtDebug>
qDebug() << "Integer value:" << 42;
qDebug() << "String value:" << myQString;
qDebug() << "Complex value:" << myQColor;
```


QDebug

DEMO

The QDebug class provides an output stream for debugging information.

1. Remember to `#include <QDebug>`
   - Use `qDebug() << "hello";`

In the common case, it is useful to call the qDebug() function to obtain a default QDebug object to use for writing debugging information.

```
qDebug() << "Date:" << QDate::currentDate();
qDebug() << "Types:" << QString("String") << QChar('x') << QRect(0, 10, 50, 40);
```

You can also write a custom:

```
QDebug operator<<(QDebug dgb, const MyClass &c);
```

To allow your own data types to be easily written to `QDebug` check the Qt documentation.