LINQ Cheat Sheet

By: Mosh Hamedani

Restriction

```
from c in context.Courses
where c.Level == 1
select c;
```

Ordering

```
from c in context.Courses
where c.Level == 1
orderby c.Name, c.Price descending
select c;
```

Projection

```
from c in context.Courses
select new { Course = c.Name, AuthorName = c.Author.Name };
```

Grouping

```
from c in context.Courses
group c by c.Level into g
select g;
```

```
from c in context.Courses
group c by c.Level into g
select new { Level = g.Key, Courses = g };
```

LINQ Cheat Sheet

By: Mosh Hamedani

Inner Join

Use when there is no relationship between your entities and you need to link them based on a key.

```
from a in context.Authors
join c in context.Courses on a.Id equals c.AuthorId
select new { Course = c.Name, Author = a.Name };
```

Group Join

Useful when you need to group objects by a property and count the number of objects in each group. In SQL we do this with LEFT JOIN, COUNT(*) and GROUP BY. In LINQ, we use group join.

```
from a in context.Authors
join c in context.Courses on a.Id equals c.AuthorId into g
select new { Author = a.Name, Courses = c.Count() };
```

Cross Join

To get full combinations of all objects on the left and the ones on the right.

```
from a in context.Authors
from c in context.Courses
select new { Author = a.Name, Course = c.Name };
```