SOLID - example

Single responsability - Email sender vs email builder. Each class has one responsability and changes to process should not bleed into the others.

Open/Closed - Open to extend closed to modification. Our system generates different types of reports. We call the mechanisms that fufill the request fore reports 'providers'. Each one requires common logic for logging and callbacks for completion so we created a ProviderBase. The base encapsolates the common functionality. So any changes to logging or callbacks only effect the base objects responsabilities and not the children.

Liskovs - subtitution - providers again (being able to swap different providers dpending on need)

Interface Segregation - I don't really have an example of interface segregation just that it's wise to keep contracts small and pointed. A class should not have to depend on an interface with signatures it has no use for. Obviusly there is a balace because we want to keep cohesion.

Dependency Inversion - This one is everywhere. I personal swapped an entire application into DI and IoC.

Angular Exp

Observalbe v promise v subject

async/blocking -

http actions

api experience

microservice experience

CORS - Prevents malicsiuse website in another tab from executing the content served up by your websites server

Continues integraiton

Continues deployment