**Asynchronous Javascript.**

**It allows you to break down bigger projects into smaller tasks.**

**Synchronous: You have ten tasks and comlete then in order, one by one.**

**Asynchronous: Three images, all loading at the same time.**

**Let’s think about a marathon. There are three competitors. If the marathons is synchronous, there is only one line. No competitor can overpass the other. Everyone must finish the race one by one. If anyone stops, the entire process stops.**

**Asynchronous marathon: Nobody stops for anyone.**

**setTimeOut allows you to run a function after a specific amount of time. setTimeOut is an asynchronous function. Other functions won’t wait until it ends to execute. The setTimeOut function is pulled to another line of execution and reléased after the specified time.**

**Callbacks: calling a function inside another function.**

**Function one(){**

**Console.log(“STEP 1”)**

**}**

**Function two(){**

**Console.log(“step 1”)**

**}**

**One()**

**Two()**

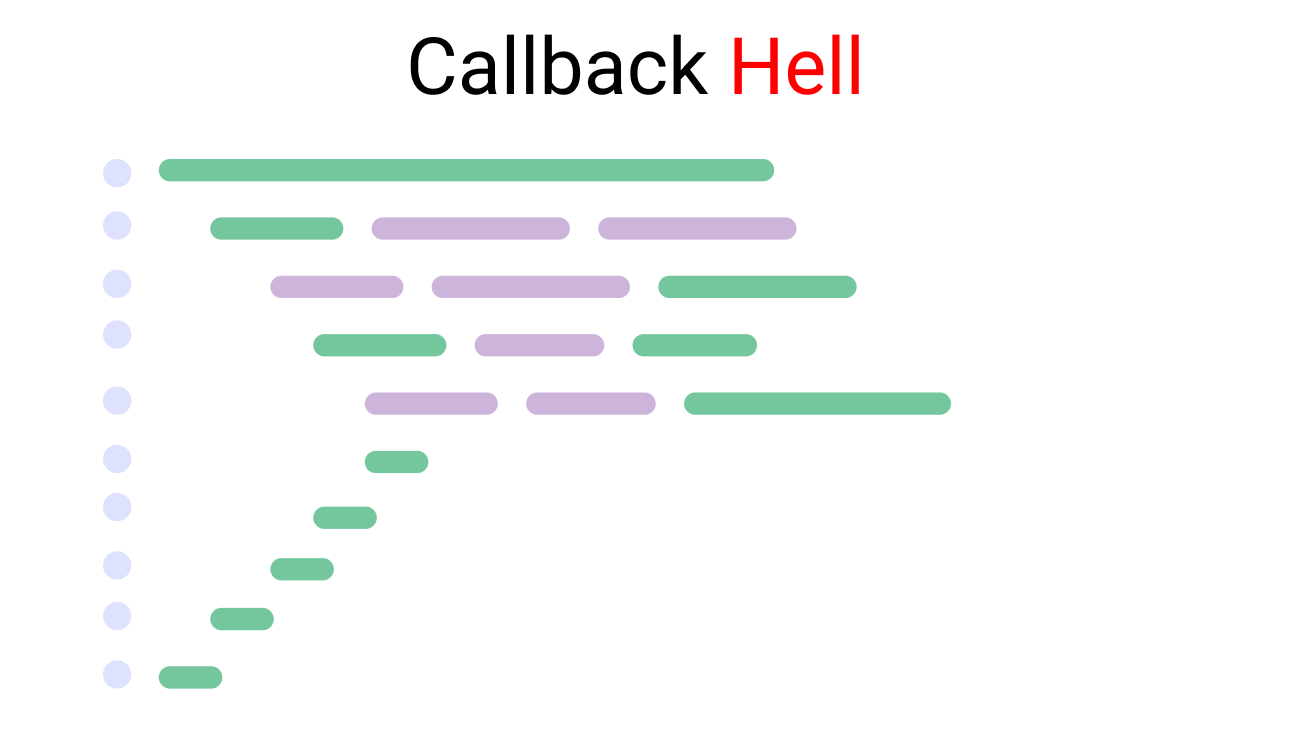
**Function one(call\_two){**

**Console.log(“One. Call function two”)**

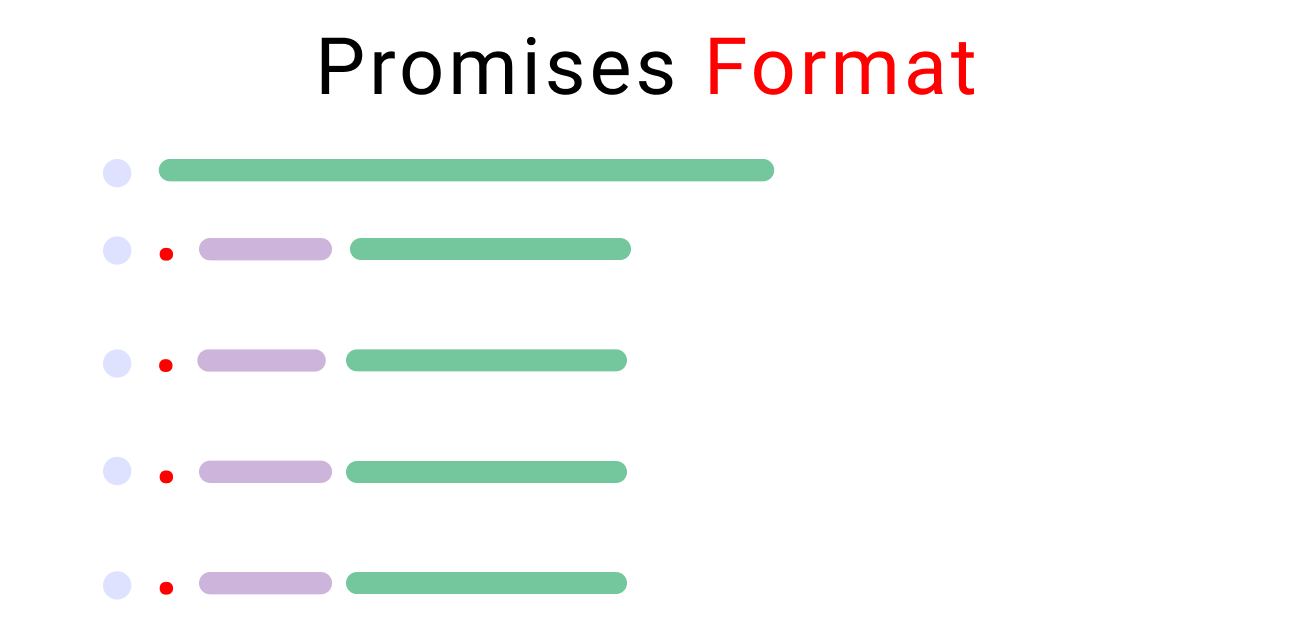
**Call\_two()**

**}**

**One(two);**



**Codigo complejo, confuso.**



**Codigo mucho mas limpio.**

**Ciclo de vida de una promesa**

****



**Callbacks make relationships. Parent, children, grandchildren, grand grandchildren, etc.**

**Promises take instruction. First do this, then this.**

**.then(chopFruit)**

**.catch((err) => {**

**console.error(err)**

**console.log(“Promise rejected”)**

**}**

**.then Works when a promise is resolved**

**.catch Works when a promise is rejected**