DEBUGGING

Topics

Topics

- Getting Started with Debugging

Topics

- Getting Started with Debugging
- Types of Bugs

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs
  - Bugs Attaching Scripts

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs
  - Bugs Attaching Scripts
  - Runtime Errors

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs
  - Bugs Attaching Scripts
  - Runtime Errors
- Stepping Through Code with the Debugger

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs
  - Bugs Attaching Scripts
  - Runtime Errors
- Stepping Through Code with the Debugger
  - Attaching the Debugger to Unity

Topics

- Getting Started with Debugging
- Types of Bugs
  - Compile-Time Bugs
  - Bugs Attaching Scripts
  - Runtime Errors
- Stepping Through Code with the Debugger
  - Attaching the Debugger to Unity
- Watching Variables in the Debugger

Getting Started with Debugging

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs
  - Track down inefficiencies

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs
  - Track down inefficiencies
- Debugging is built in to Unity via MonoDevelop

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs
  - Track down inefficiencies
- Debugging is built in to Unity via MonoDevelop
  - The MonoDevelop debugger can *attach* to the Unity process to debug your code

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs
  - Track down inefficiencies
- Debugging is built in to Unity via MonoDevelop
  - The MonoDevelop debugger can *attach* to the Unity process to debug your code
  - And, the MonoDevelop debugger can connect to an iOS or Android device and debug code running on the device!!!

Getting Started with Debugging

- **Debugging is a way to step through and watch your code as it is running**
- **This can help you**
  - **Better understand code**
  - **Find errors and bugs**
  - **Track down inefficiencies**
- **Debugging is built in to Unity via MonoDevelop**
  - **The MonoDevelop debugger can *attach* to the Unity process to debug your code**
  - **And, the MonoDevelop debugger can connect to an iOS or Android device and debug code running on the device!!!**
    - This is **very** helpful for finding issues with touch interfaces

Getting Started with Debugging

- **Debugging is a way to step through and watch your code as it is running**
- **This can help you**
  - **Better understand code**
  - **Find errors and bugs**
  - **Track down inefficiencies**
- **Debugging is built in to Unity via MonoDevelop**
  - **The MonoDevelop debugger can *attach* to the Unity process to debug your code**
  - **And, the MonoDevelop debugger can connect to an iOS or Android device and debug code running on the device!!!**
    - This is **very** helpful for finding issues with touch interfaces
    - Can be done over either a cable or WiFi!

Getting Started with Debugging

- Debugging is a way to step through and watch your code as it is running
- This can help you
  - Better understand code
  - Find errors and bugs
  - Track down inefficiencies
- Debugging is built in to Unity via MonoDevelop
  - The MonoDevelop debugger can *attach* to the Unity process to debug your code
  - And, the MonoDevelop debugger can connect to an iOS or Android device and debug code running on the device!!!
    - This is **very** helpful for finding issues with touch interfaces
    - Can be done over either a cable or WiFi!
  - The book has detailed instructions for using the debugger

Types of Bugs

Types of Bugs

- Compile-Time Bugs

Types of Bugs

- **Compile-Time Bugs**
  - A bug found in the syntax of your code

Types of Bugs

- **Compile-Time Bugs**
  - A bug found in the syntax of your code
  - Compile-time bugs prevent your code from compiling

Types of Bugs

- **Compile-Time Bugs**
  - **A bug found in the syntax of your code**
  - **Compile-time bugs prevent your code from compiling**
    - Makes it unusable in Unity until the bug is resolved

Types of Bugs

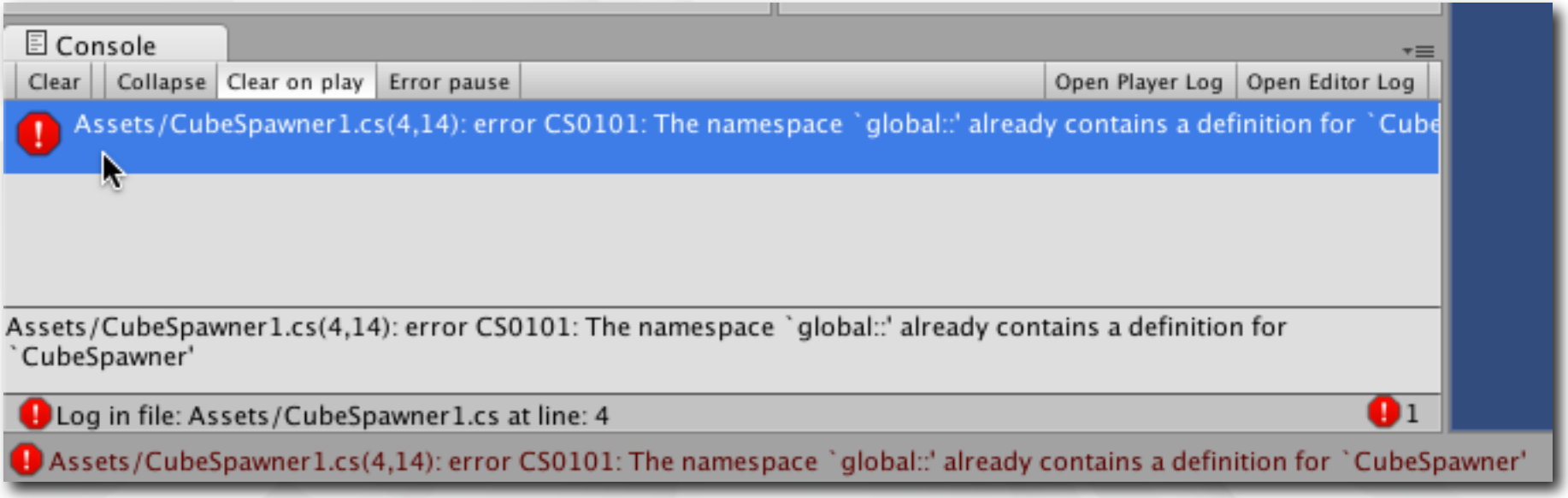
- **Compile-Time Bugs**
  - **A bug found in the syntax of your code**
  - **Compile-time bugs prevent your code from compiling**
    - Makes it unusable in Unity until the bug is resolved
  - **Compile-time bugs usually cause very specific errors**

Types of Bugs

- **Compile-Time Bugs**
  - **A bug found in the syntax of your code**
  - **Compile-time bugs prevent your code from compiling**
    - Makes it unusable in Unity until the bug is resolved
  - **Compile-time bugs usually cause very specific errors**
    - The error below is on line 4, character 14 of CubeSpawner1.cs

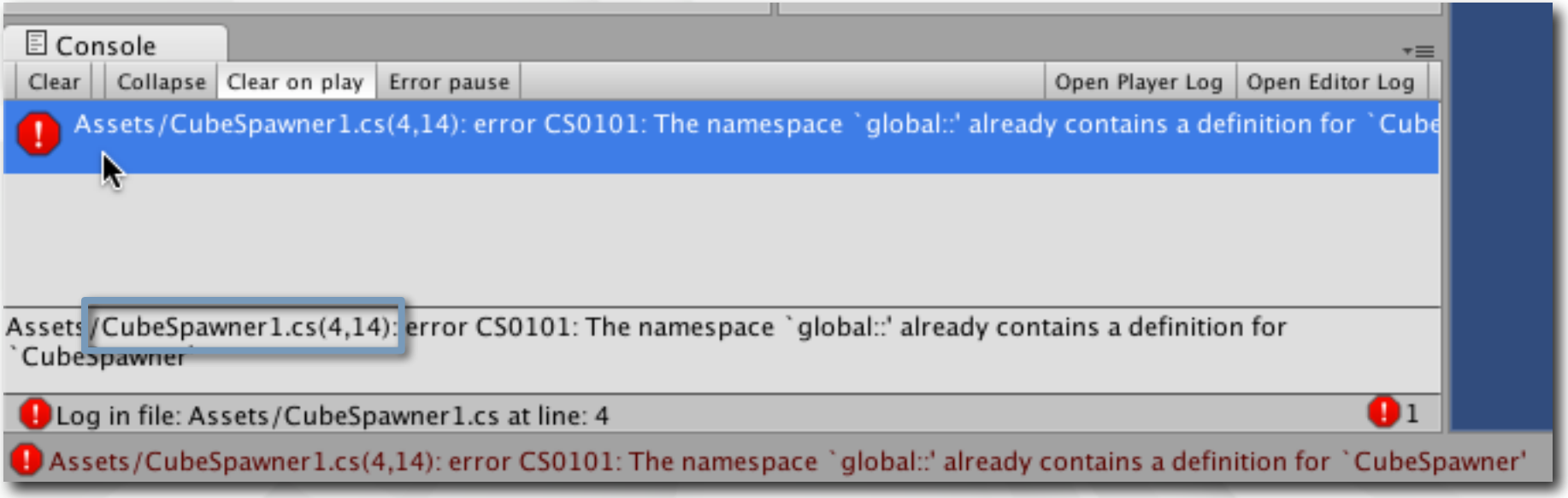
Types of Bugs

- **Compile-Time Bugs**
  - **A bug found in the syntax of your code**
  - **Compile-time bugs prevent your code from compiling**
    - Makes it unusable in Unity until the bug is resolved
  - **Compile-time bugs usually cause very specific errors**
    - The error below is on line 4, character 14 of CubeSpawner1.cs

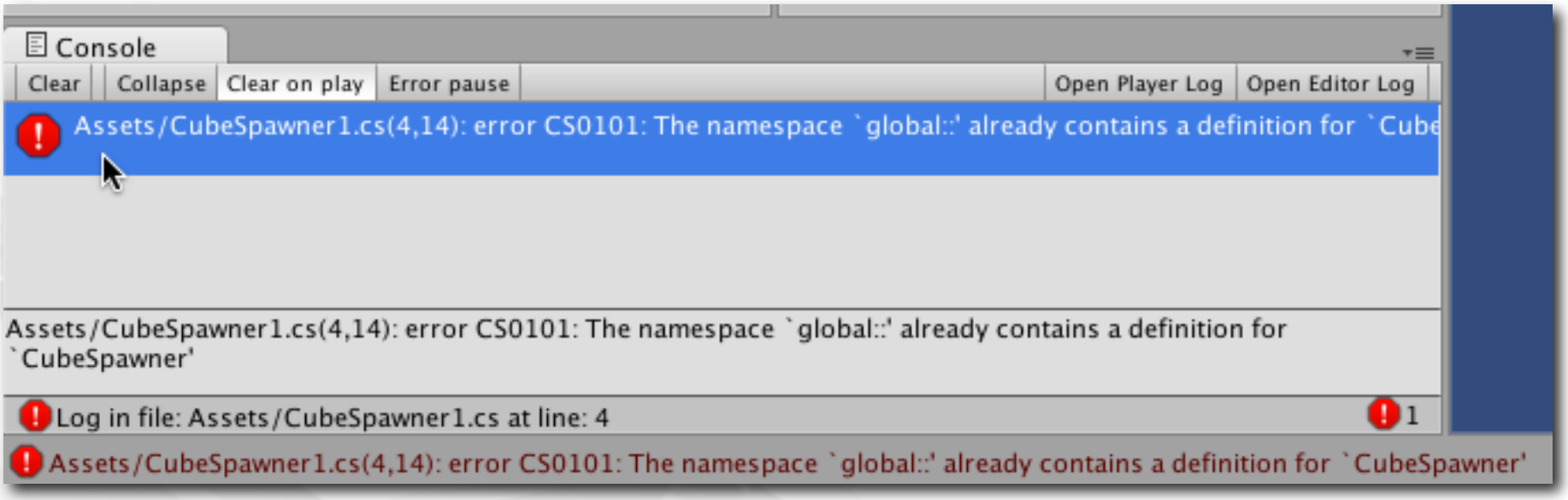


Types of Bugs

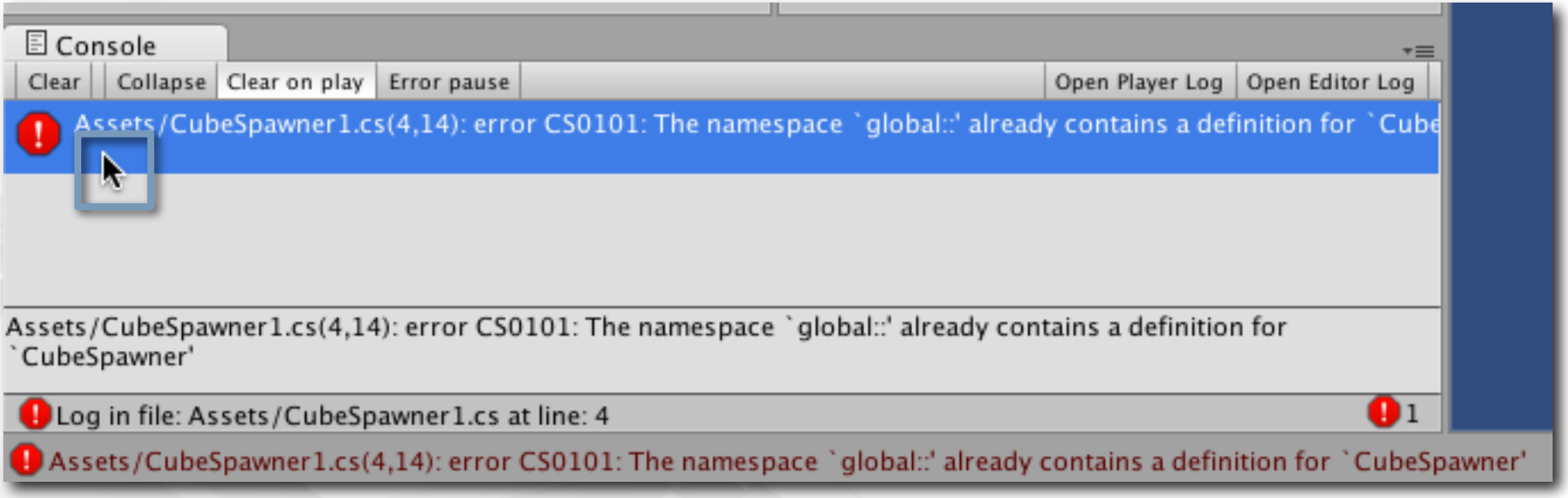
- **Compile-Time Bugs**
  - **A bug found in the syntax of your code**
  - **Compile-time bugs prevent your code from compiling**
    - Makes it unusable in Unity until the bug is resolved
  - **Compile-time bugs usually cause very specific errors**
    - The error below is on line 4, character 14 of CubeSpawner1.cs



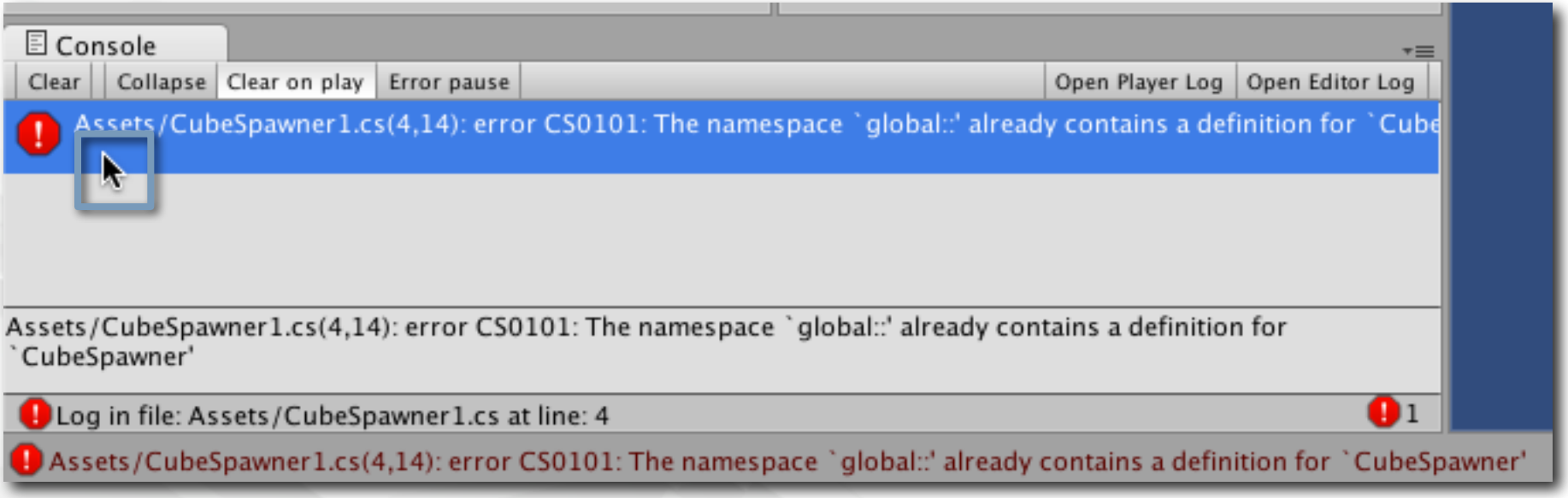
Anatomy of a Compile-Time Bug



Anatomy of a Compile-Time Bug

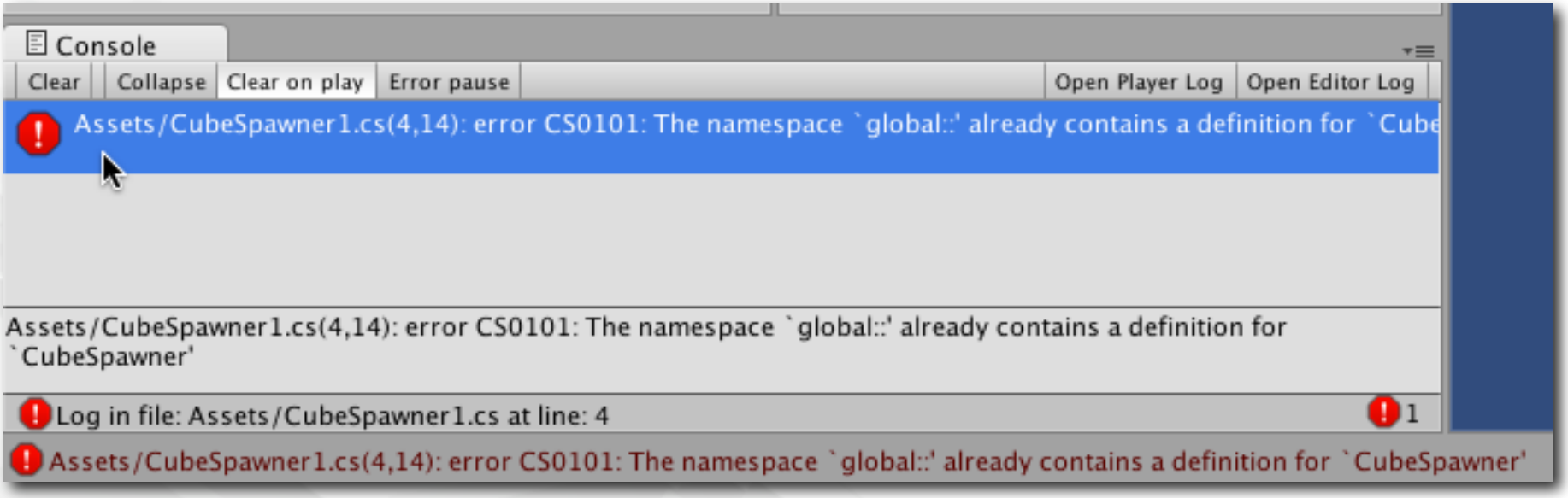


Anatomy of a Compile-Time Bug



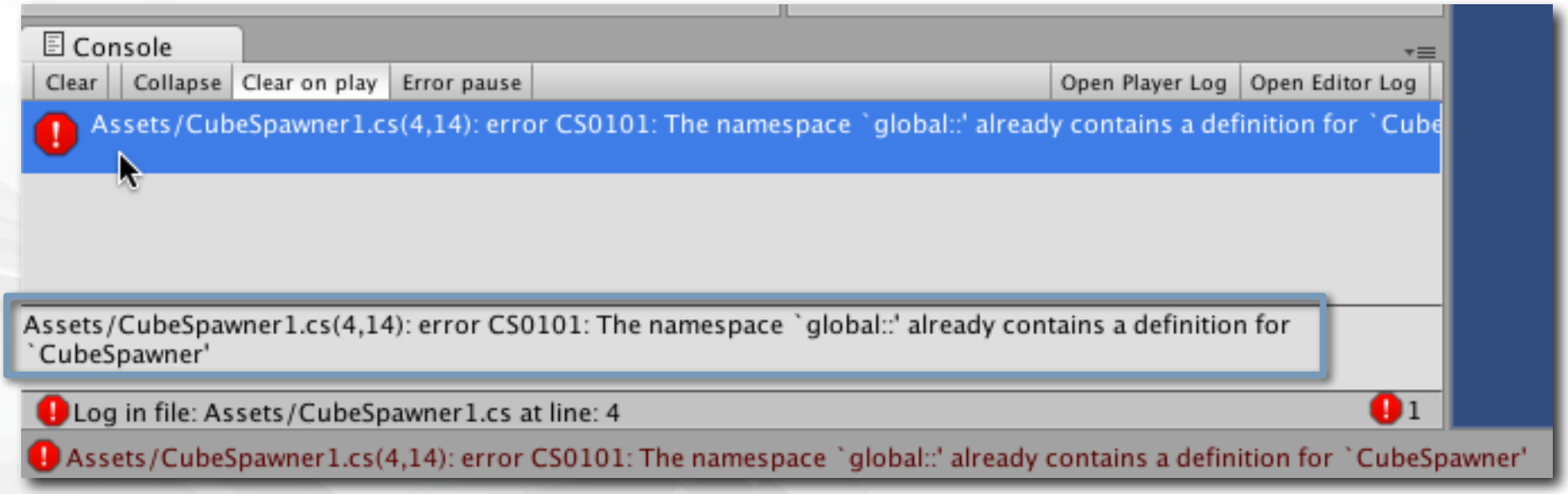
- Click the error message to get more information

Anatomy of a Compile-Time Bug



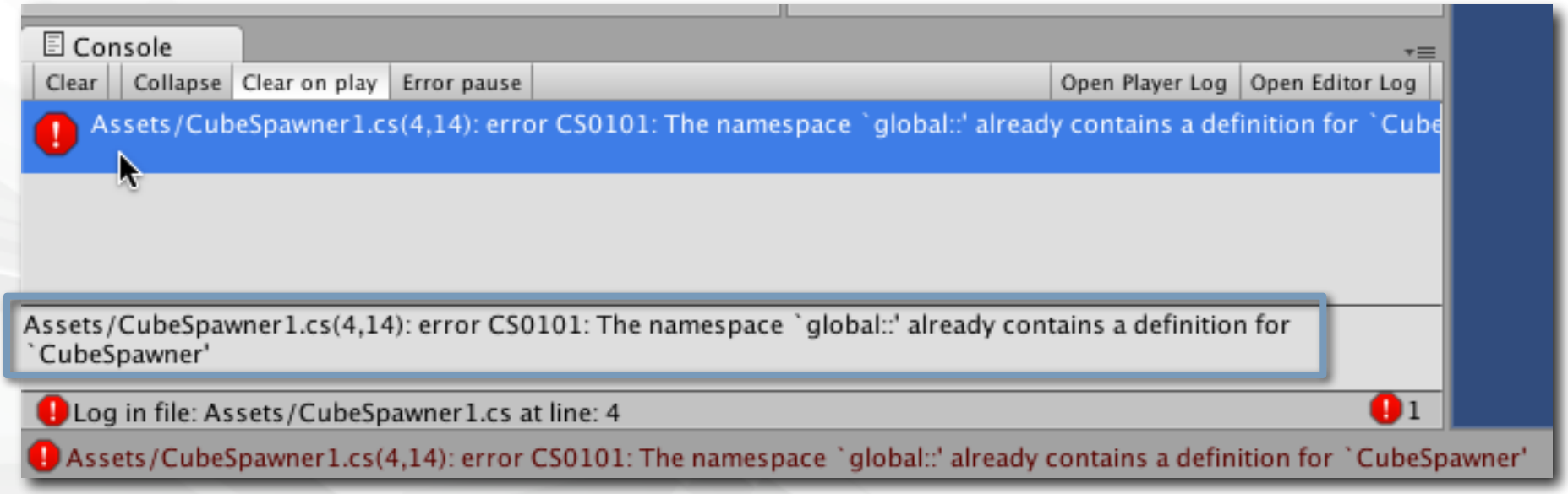
- Click the error message to get more information

Anatomy of a Compile-Time Bug



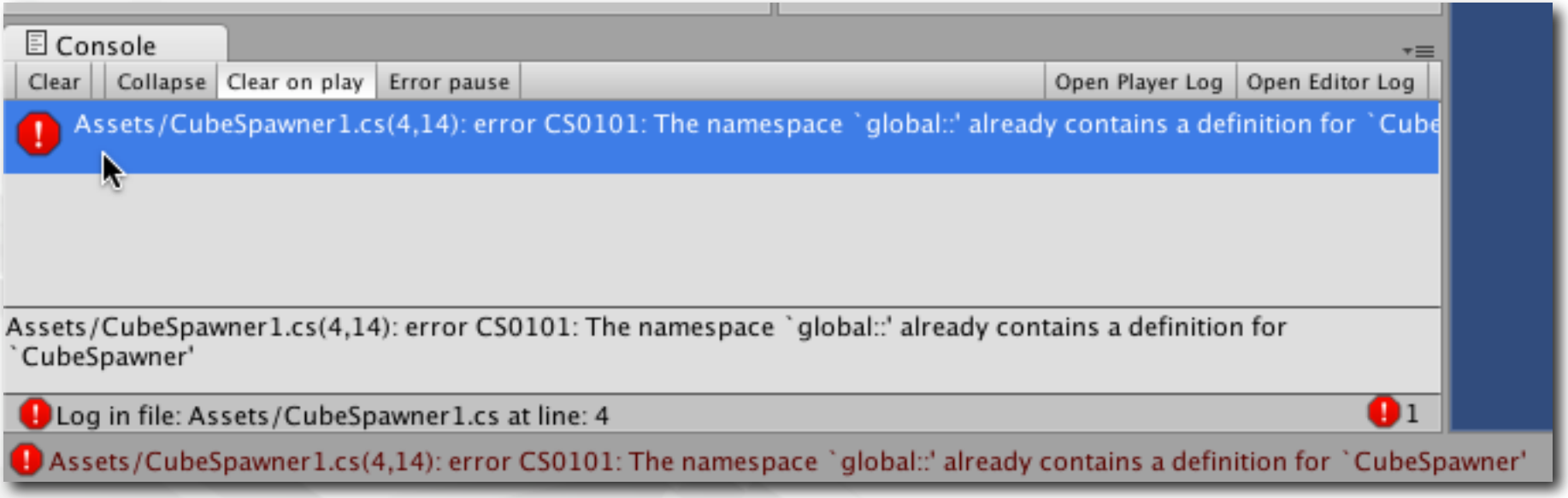
- Click the error message to get more information

Anatomy of a Compile-Time Bug



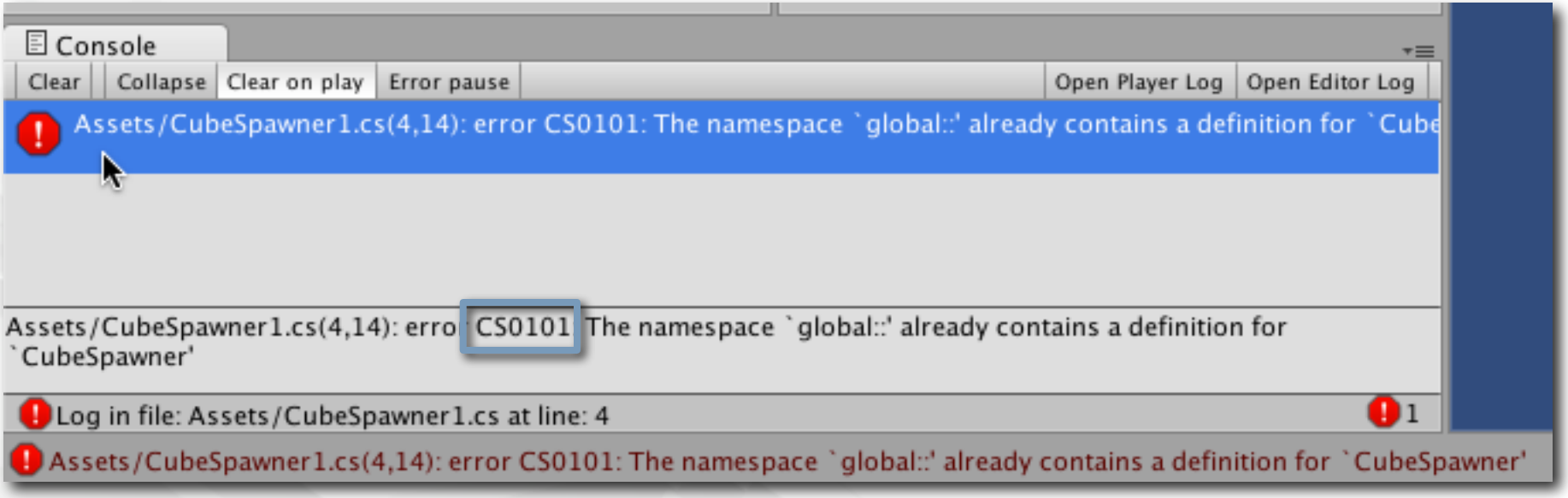
- Click the error message to get more information
- The full error text can usually tell you what's wrong

Anatomy of a Compile-Time Bug



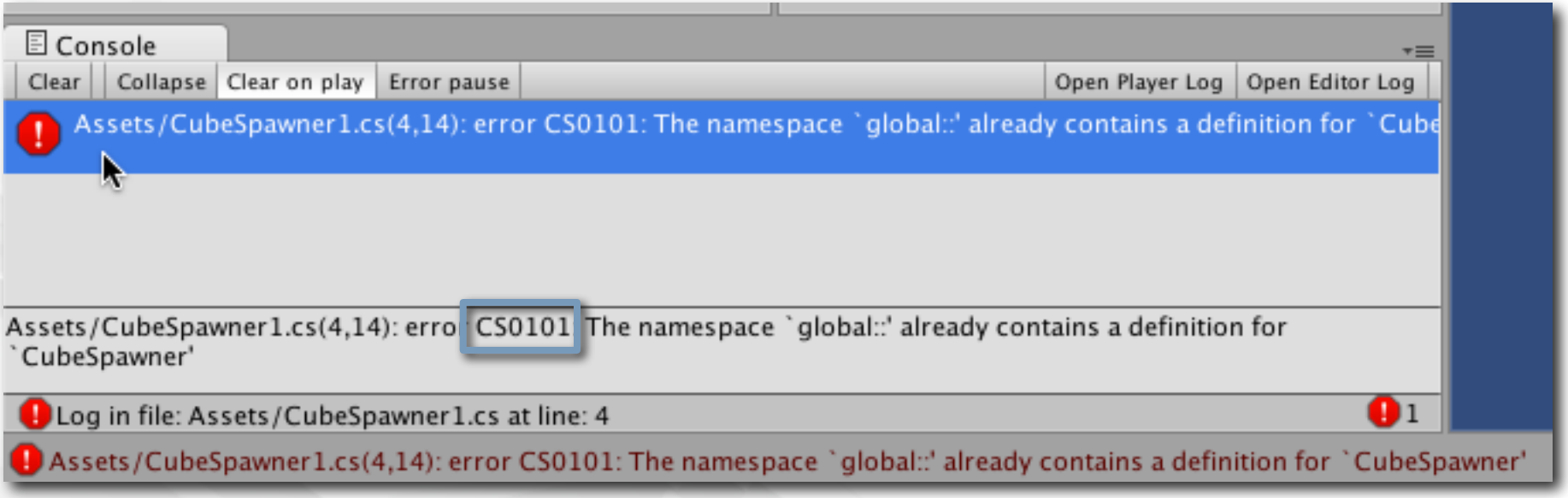
- Click the error message to get more information
- The full error text can usually tell you what's wrong

Anatomy of a Compile-Time Bug



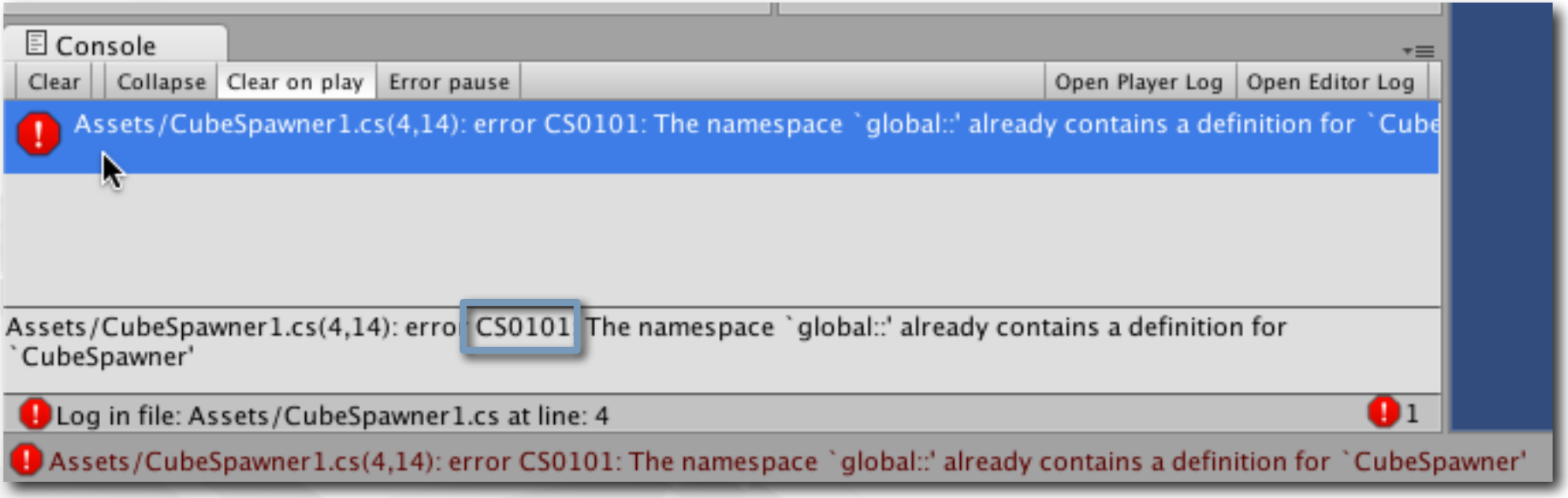
- Click the error message to get more information
- The full error text can usually tell you what's wrong

Anatomy of a Compile-Time Bug



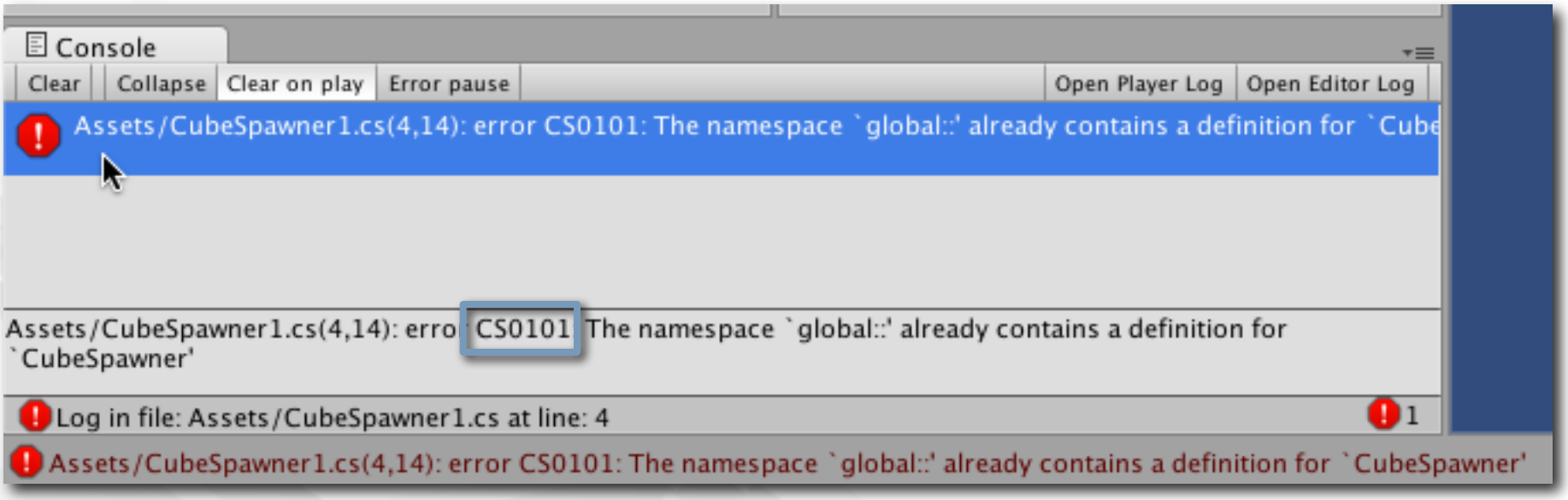
- Click the error message to get more information
- The full error text can usually tell you what's wrong
  - If not, search the Internet for the error number

Anatomy of a Compile-Time Bug



- Click the error message to get more information
- The full error text can usually tell you what's wrong
  - If not, search the Internet for the error number
  - Example: "Unity error CS0101"

Anatomy of a Compile-Time Bug



- **Click the error message to get more information**
- **The full error text can usually tell you what's wrong**
  - **If not, search the Internet for the error number**
  - **Example: "Unity error CS0101"**
    - Unity forums and StackOverflow.com have some of the best answers

Common Compile-Time Errors to Know

Common Compile-Time Errors to Know

- error CS0101: The namespace 'global::' already contains a definition for '_____'

Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**

Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**
    - Change the name of the class in one of the scripts

Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**
    - Change the name of the class in one of the scripts
- **error CS1525: Unexpected symbol '}'**

Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**
    - Change the name of the class in one of the scripts
- **error CS1525: Unexpected symbol '}'**
  - **Many "Unexpected symbol" errors are caused by a semicolon missing on a previous line or a misplaced brace**

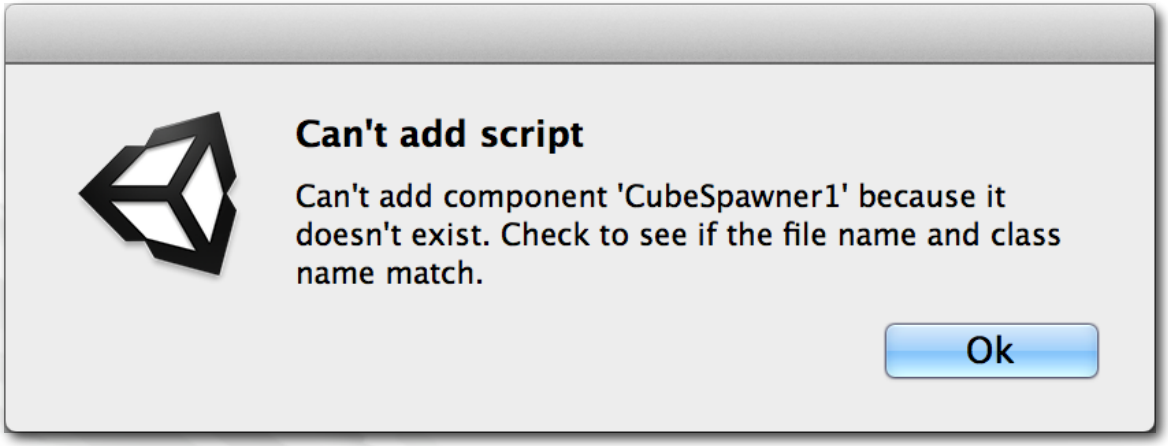
Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**
    - Change the name of the class in one of the scripts
- **error CS1525: Unexpected symbol '}'**
  - **Many "Unexpected symbol" errors are caused by a semicolon missing on a previous line or a misplaced brace**
    - Check line endings for semicolons **;**

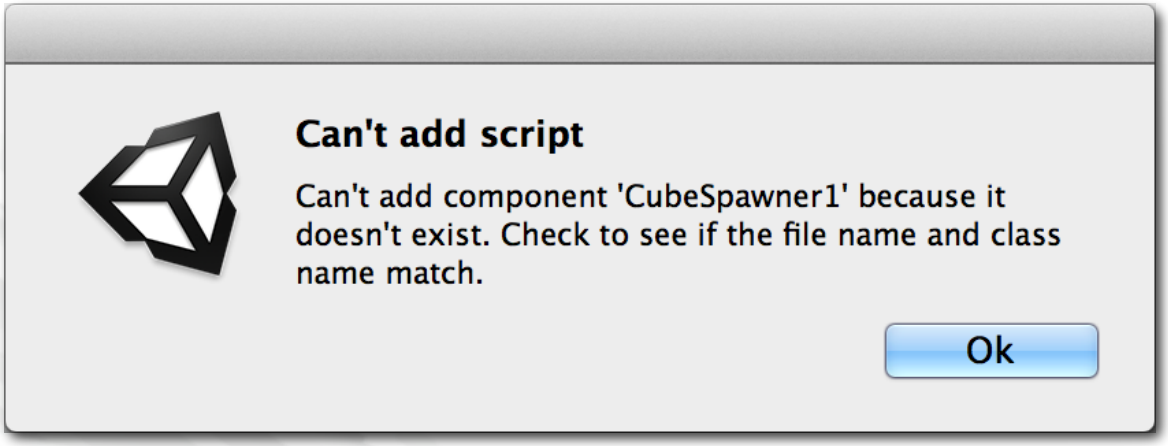
Common Compile-Time Errors to Know

- **error CS0101: The namespace 'global::' already contains a definition for '_____'**
  - **Two scripts are trying to define the same class**
    - Change the name of the class in one of the scripts
- **error CS1525: Unexpected symbol '}'**
  - **Many "Unexpected symbol" errors are caused by a semicolon missing on a previous line or a misplaced brace**
    - Check line endings for semicolons **;**
    - Check to make sure all braces have a mate **{ }**

Bugs Attaching Scripts to GameObjects

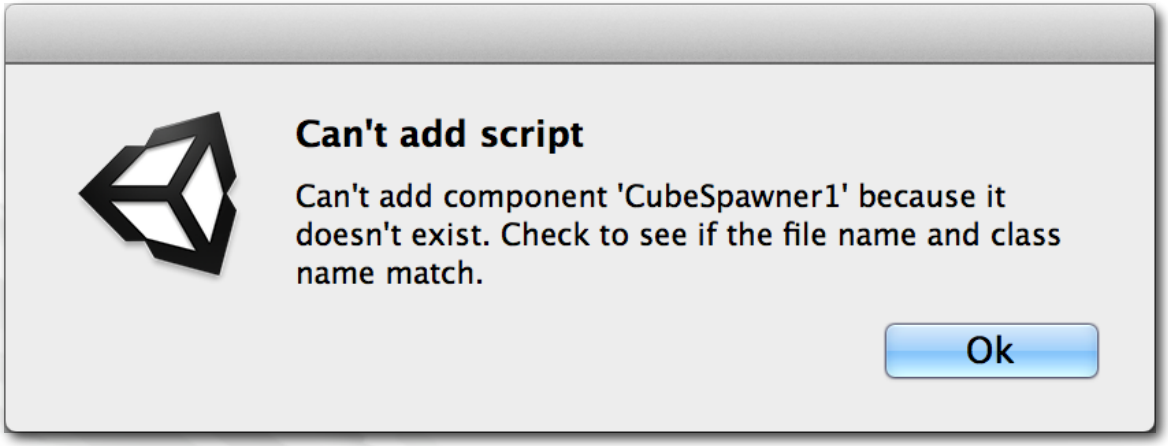


Bugs Attaching Scripts to GameObjects



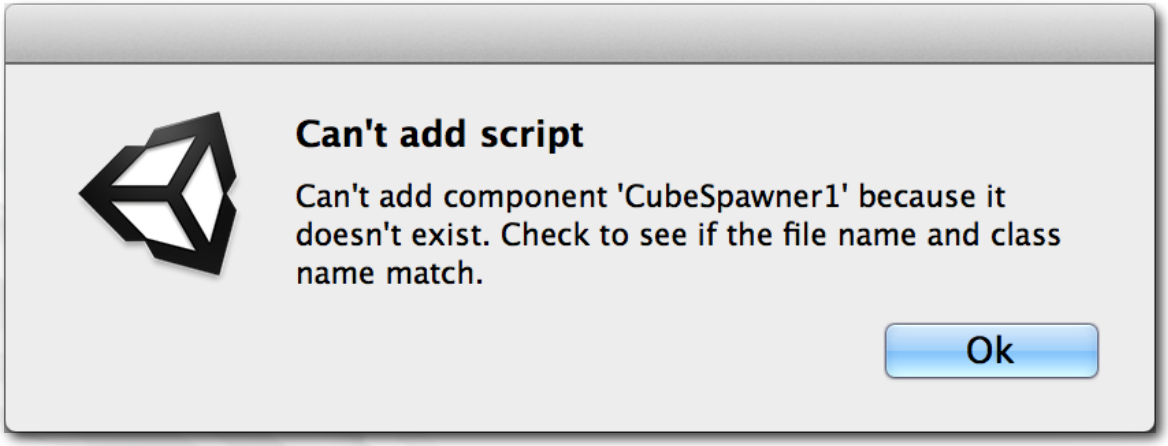
- Error occurs when attempting to attach a script to a GameObject

Bugs Attaching Scripts to GameObjects



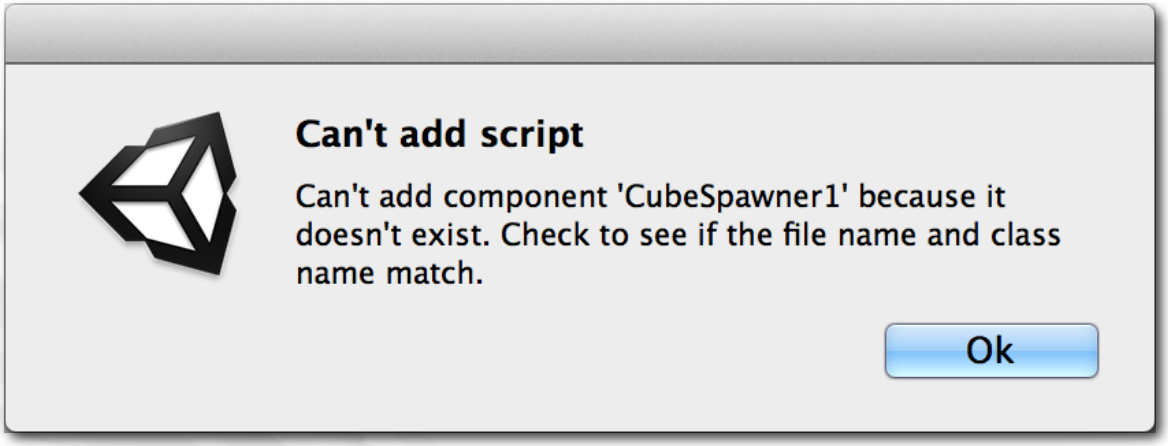
- **Error occurs when attempting to attach a script to a GameObject**
  - **Caused by the name of the script not matching the name of the defined class**

Bugs Attaching Scripts to GameObjects



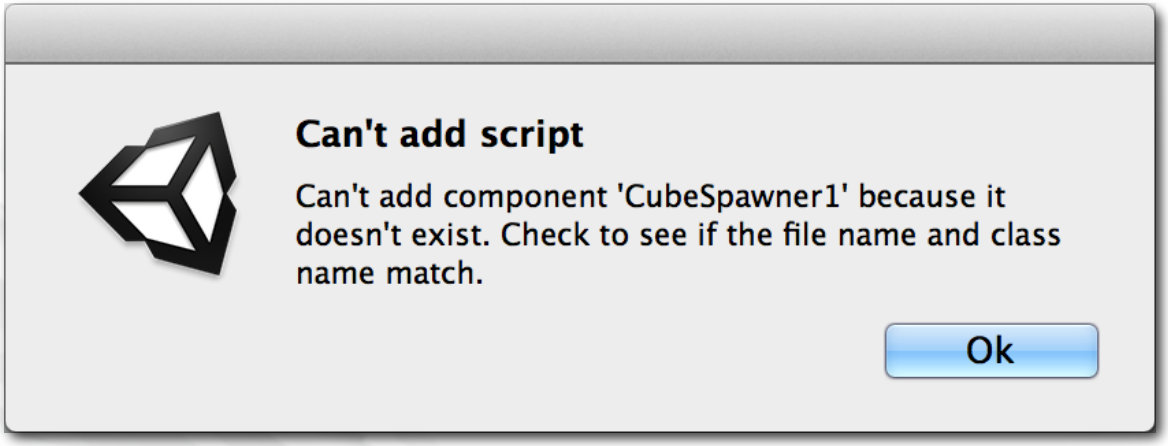
- Error occurs when attempting to attach a script to a GameObject
  - Caused by the name of the script not matching the name of the defined class
- Example

Bugs Attaching Scripts to GameObjects



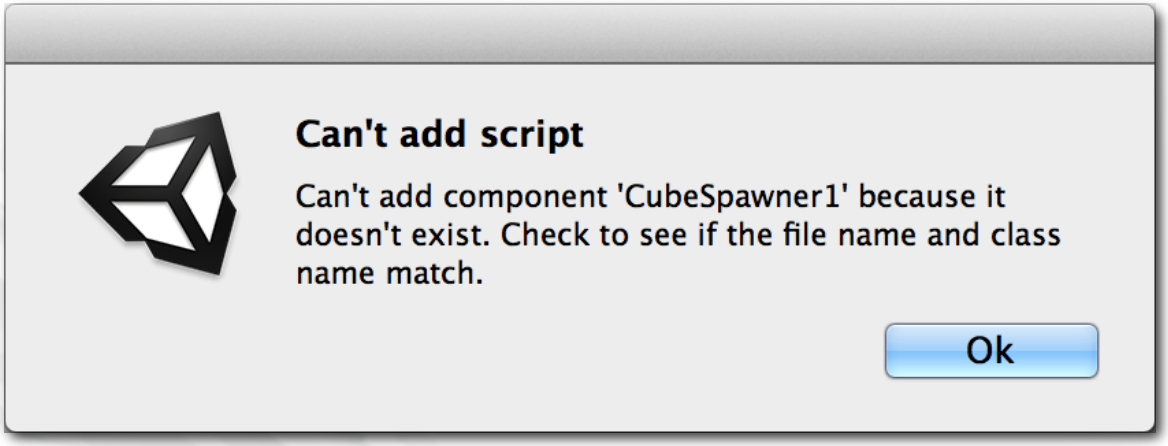
- **Error occurs when attempting to attach a script to a GameObject**
  - **Caused by the name of the script not matching the name of the defined class**
- **Example**
  - **Script filename: CubeSpawner1** (or CubeSpawner1.cs)

Bugs Attaching Scripts to GameObjects



- **Error occurs when attempting to attach a script to a GameObject**
  - **Caused by the name of the script not matching the name of the defined class**
- **Example**
  - **Script filename: CubeSpawner1** (or CubeSpawner1.cs)
  - **Class name:** `public class CubeSpawner : MonoBehaviour { … }`

Bugs Attaching Scripts to GameObjects



- **Error occurs when attempting to attach a script to a GameObject**
  - **Caused by the name of the script not matching the name of the defined class**
- **Example**
  - **Script filename: CubeSpawner1** (or CubeSpawner1.cs)
  - **Class name:** `public class CubeSpawner : MonoBehaviour { … }`
- **To Fix: Match the names to each other**

Types of Bugs

Types of Bugs

- Runtime Errors

Types of Bugs

- **Runtime Errors**
  - A bug that occurs when your code is running

Types of Bugs

- **Runtime Errors**
  - A bug that occurs when your code is running
  - Unity has no way of predicting these

Types of Bugs

- **Runtime Errors**
  - A bug that occurs when your code is running
  - Unity has no way of predicting these
- **Most common types of Runtime Errors**

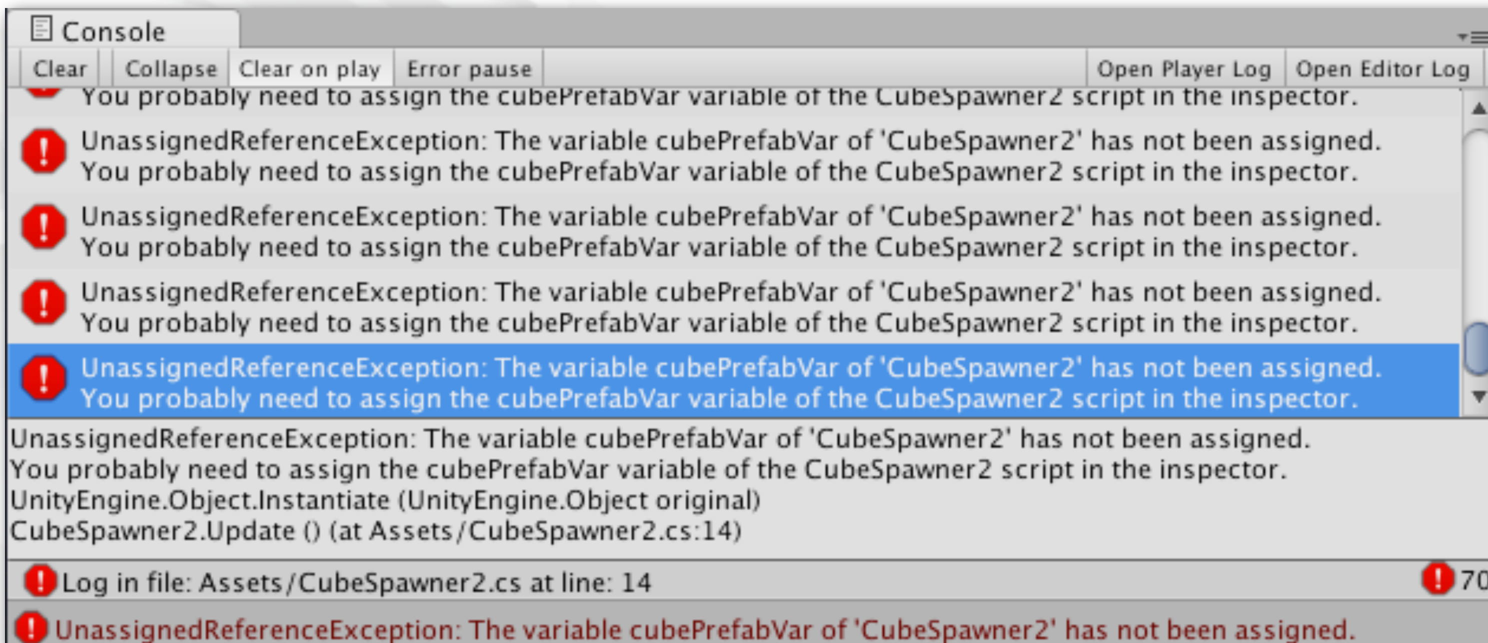
Types of Bugs

- **Runtime Errors**
  - A bug that occurs when your code is running
  - Unity has no way of predicting these
- **Most common types of Runtime Errors**
  - UnassignedReferenceException

Types of Bugs

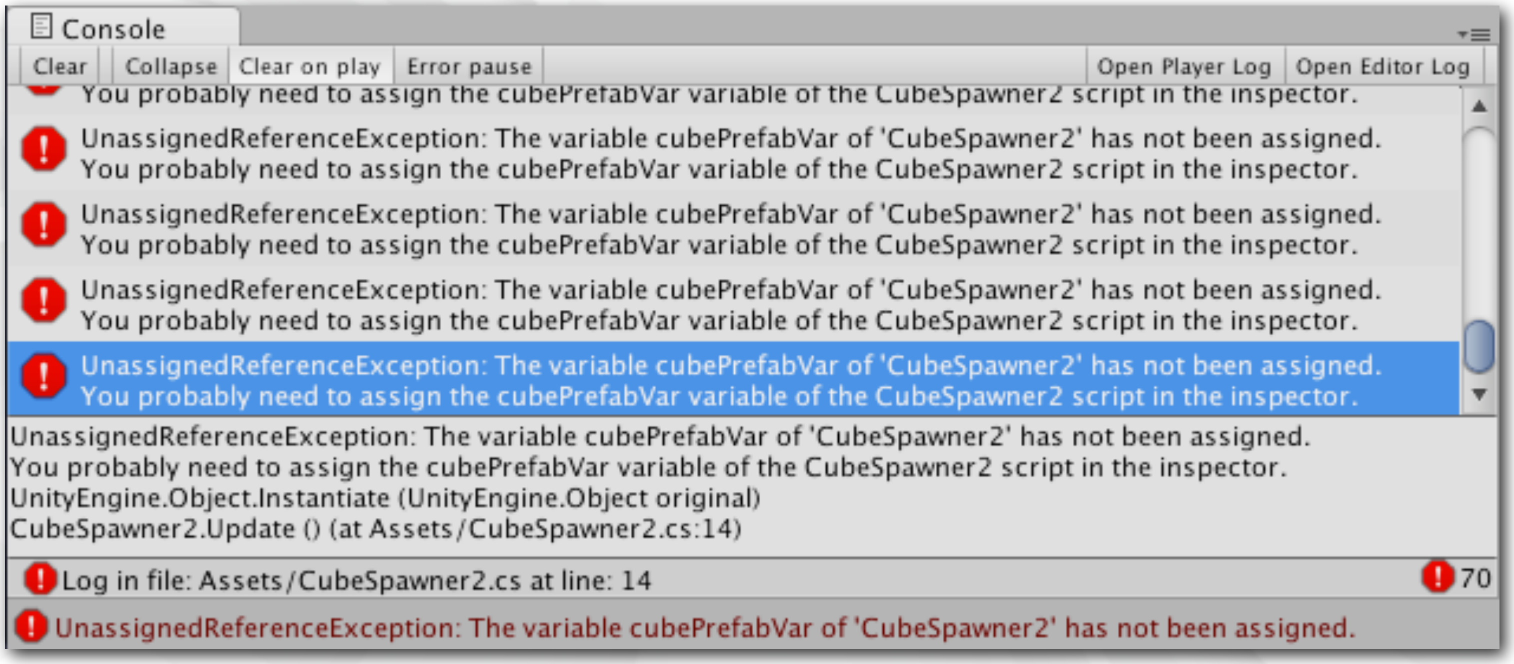
- **Runtime Errors**
  - A bug that occurs when your code is running
  - Unity has no way of predicting these
- **Most common types of Runtime Errors**
  - UnassignedReferenceException
  - NullReferenceException

Common Runtime Errors



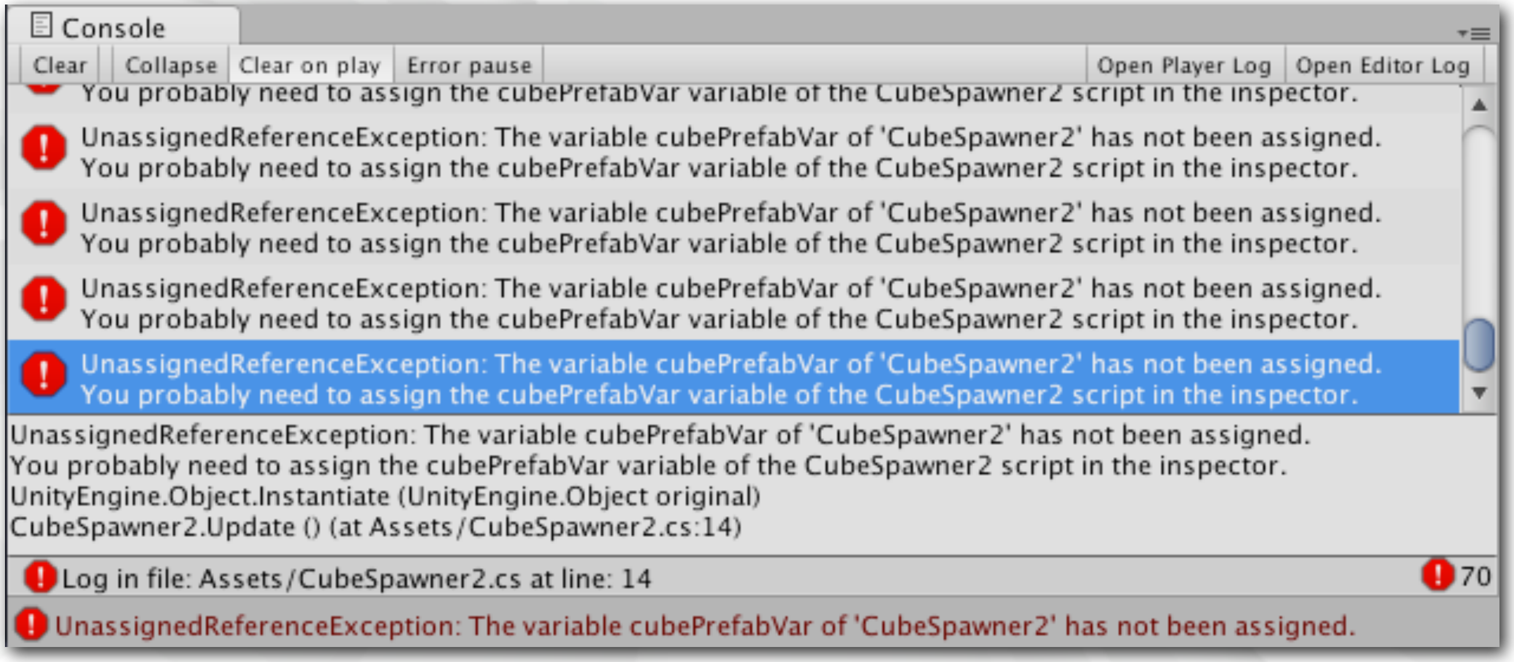
Common Runtime Errors

- UnassignedReferenceException



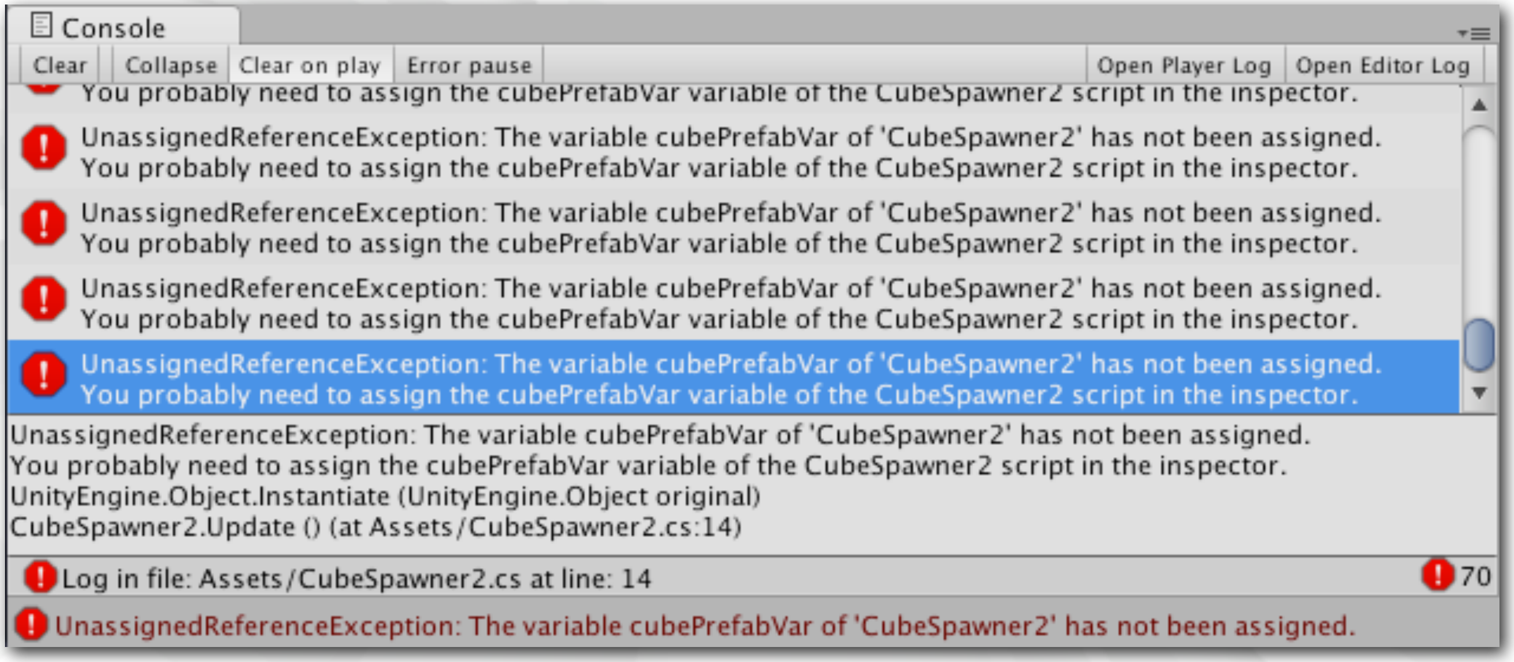
Common Runtime Errors

- **UnassignedReferenceException**
  - A variable in the Inspector has not been set



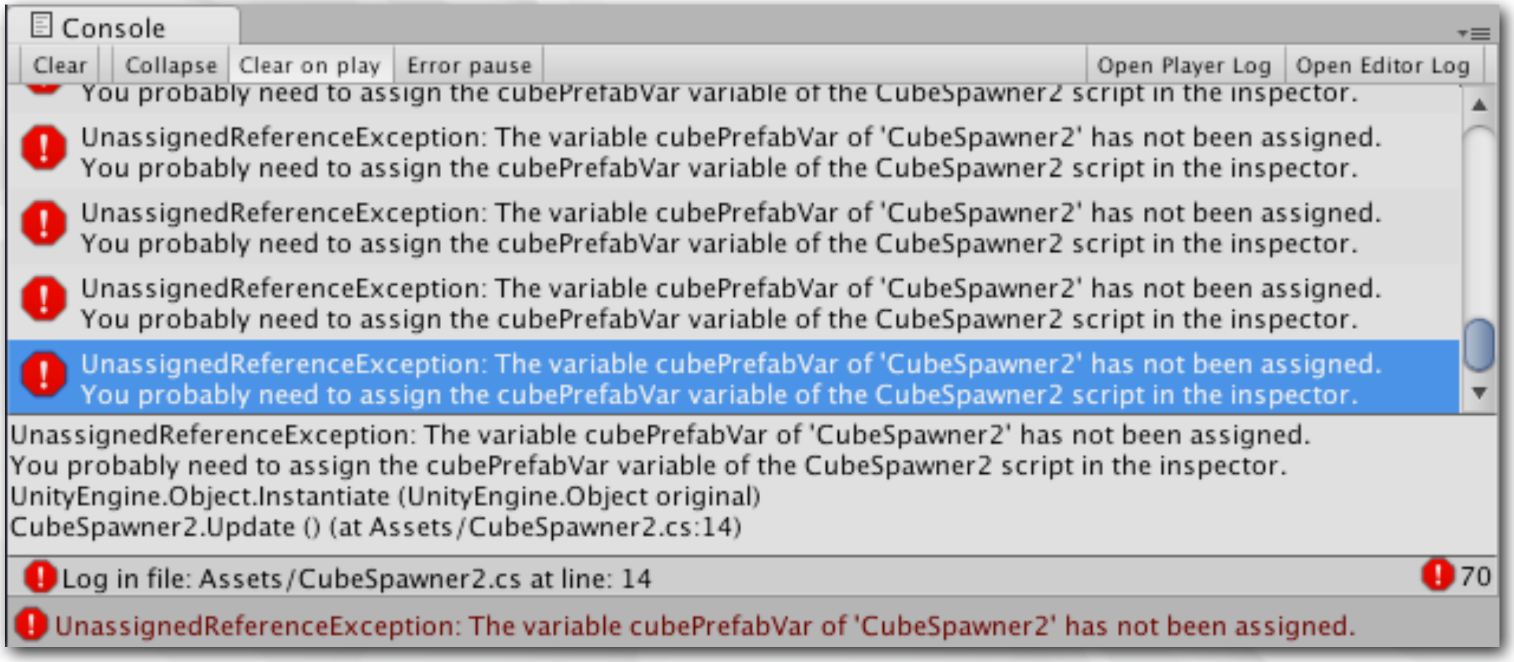
Common Runtime Errors

- **UnassignedReferenceException**
  - **A variable in the Inspector has not been set**
    - Most commonly GameObject prefabs for Instantiate() calls



Common Runtime Errors

- **UnassignedReferenceException**
  - **A variable in the Inspector has not been set**
    - Most commonly GameObject prefabs for Instantiate() calls
  - **To Fix: Assign the variable in the Inspector**



Common Runtime Errors

Common Runtime Errors

- Null Reference Exception

Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist

Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

Common Runtime Errors

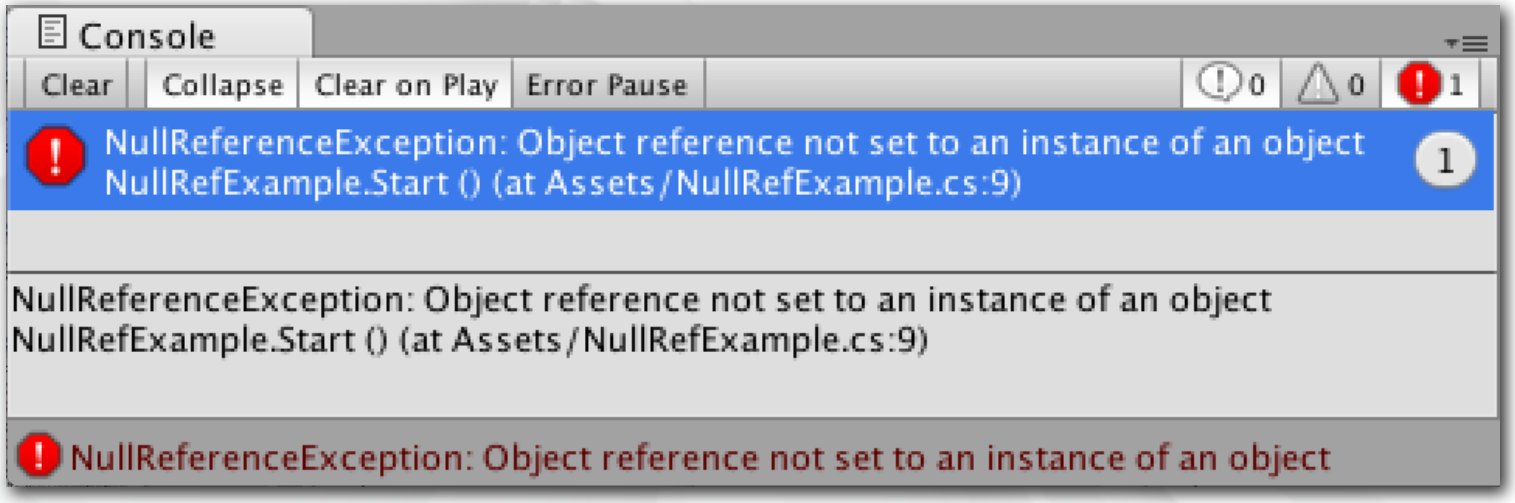
- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

```
 7    void Start () {
 8       GameObject[] goArray = new GameObject[10];
 9       print (goArray[5].transform.position);
10    } // on line 9, goArray[5] is null, so it has no transform
```

Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

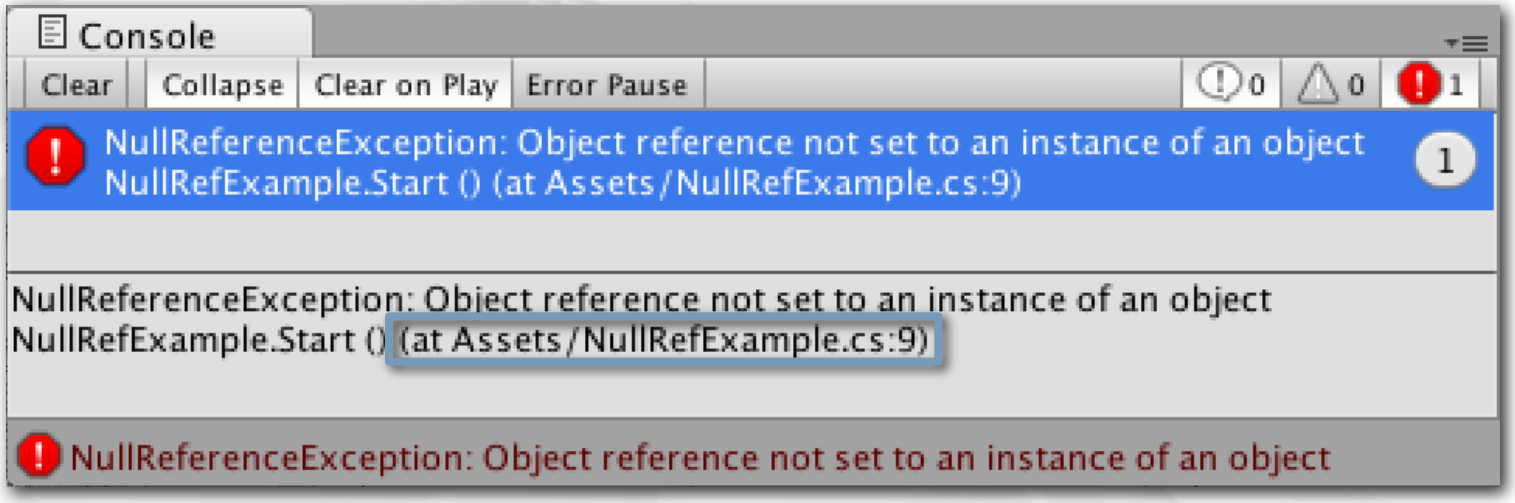
```
 7    void Start () {
 8        GameObject[] goArray = new GameObject[10];
 9        print (goArray[5].transform.position);
10    } // on line 9, goArray[5] is null, so it has no transform
```



Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

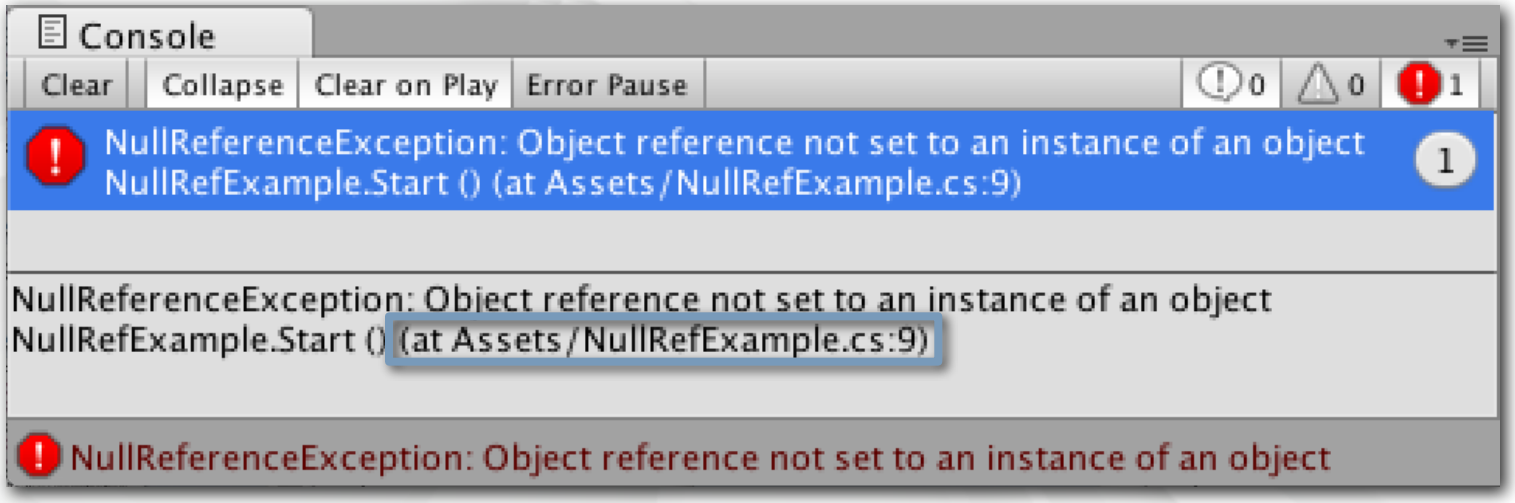
```
 7    void Start () {
 8        GameObject[] goArray = new GameObject[10];
 9        print (goArray[5].transform.position);
10    } // on line 9, goArray[5] is null, so it has no transform
```



Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

```
 7    void Start () {
 8        GameObject[] goArray = new GameObject[10];
 9        print (goArray[5].transform.position);
10    } // on line 9, goArray[5] is null, so it has no transform
```

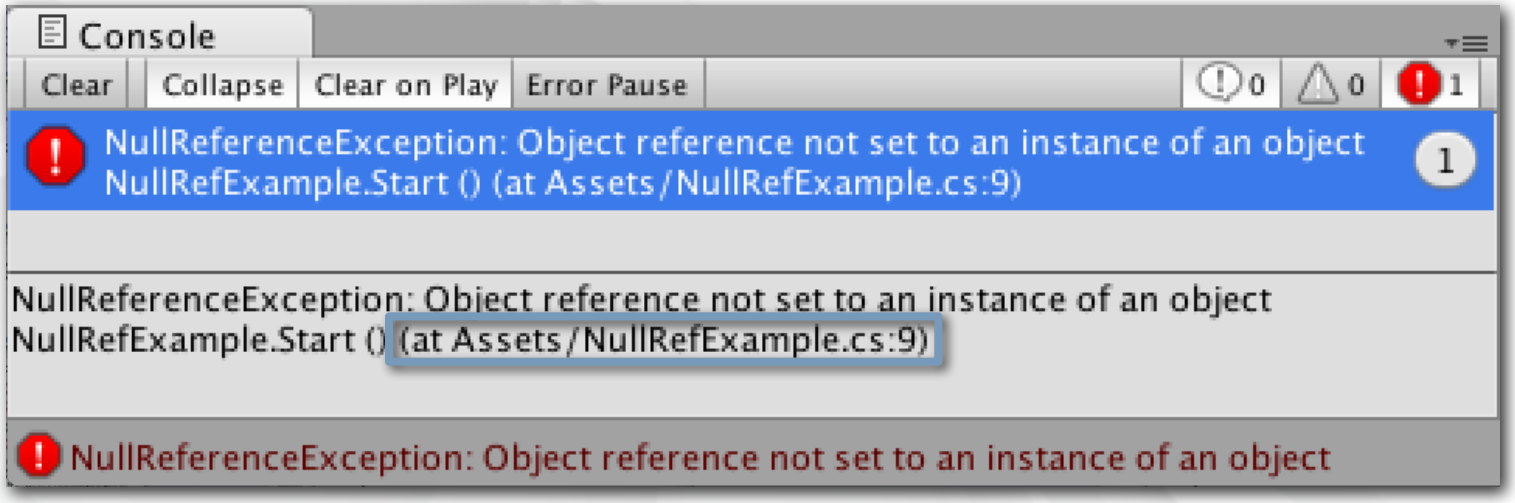


  - Error can only tell you the line number

Common Runtime Errors

- **Null Reference Exception**
  - Unity has been asked to access something that doesn't exist
  - Example:

```
 7    void Start () {
 8        GameObject[] goArray = new GameObject[10];
 9        print (goArray[5].transform.position);
10    } // on line 9, goArray[5] is null, so it has no transform
```



  - Error can only tell you the line number
  - These are difficult to debug!

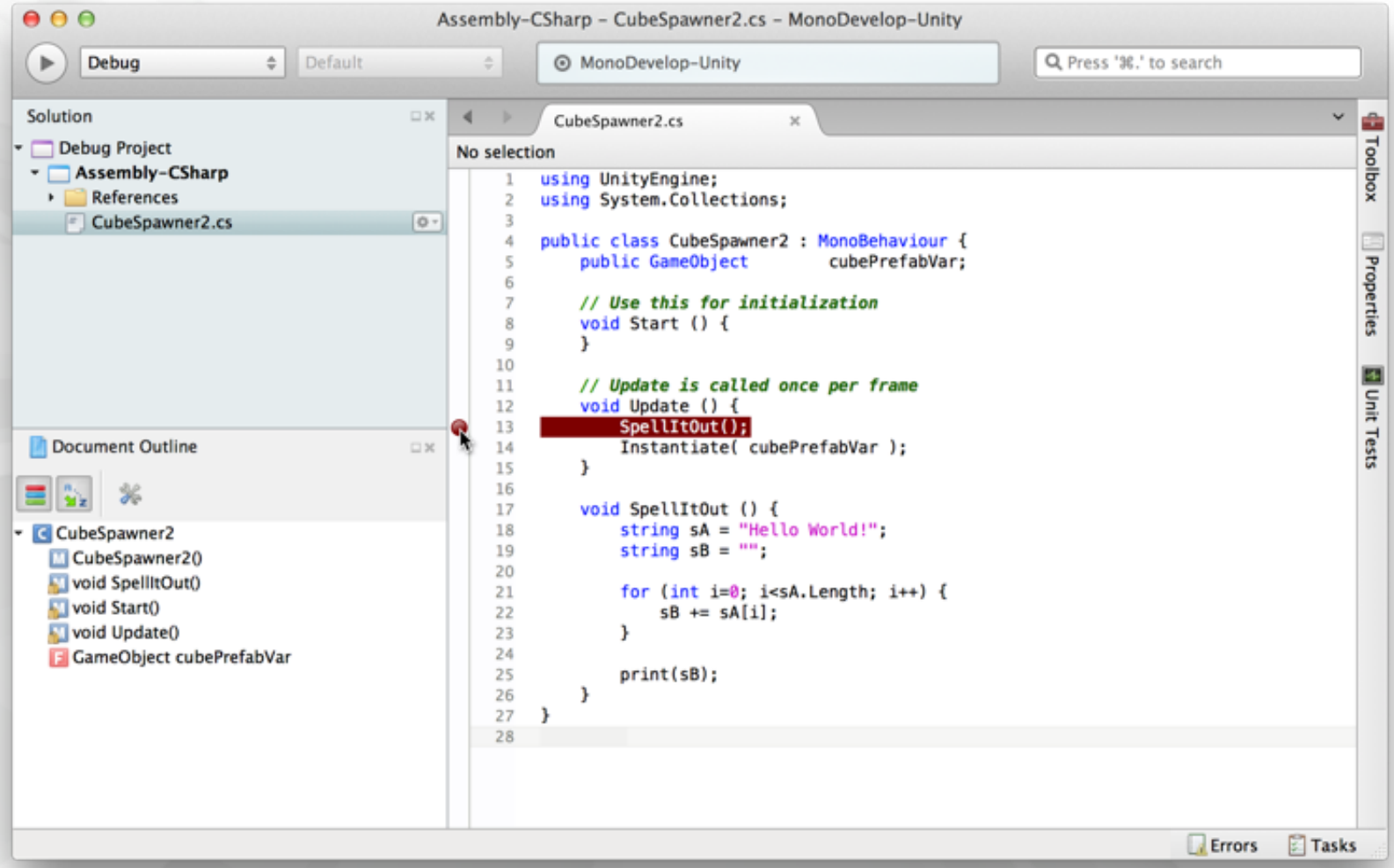
Stepping Through Code with the Debugger

Stepping Through Code with the Debugger

- **Step 1: Set a Breakpoint in your code**

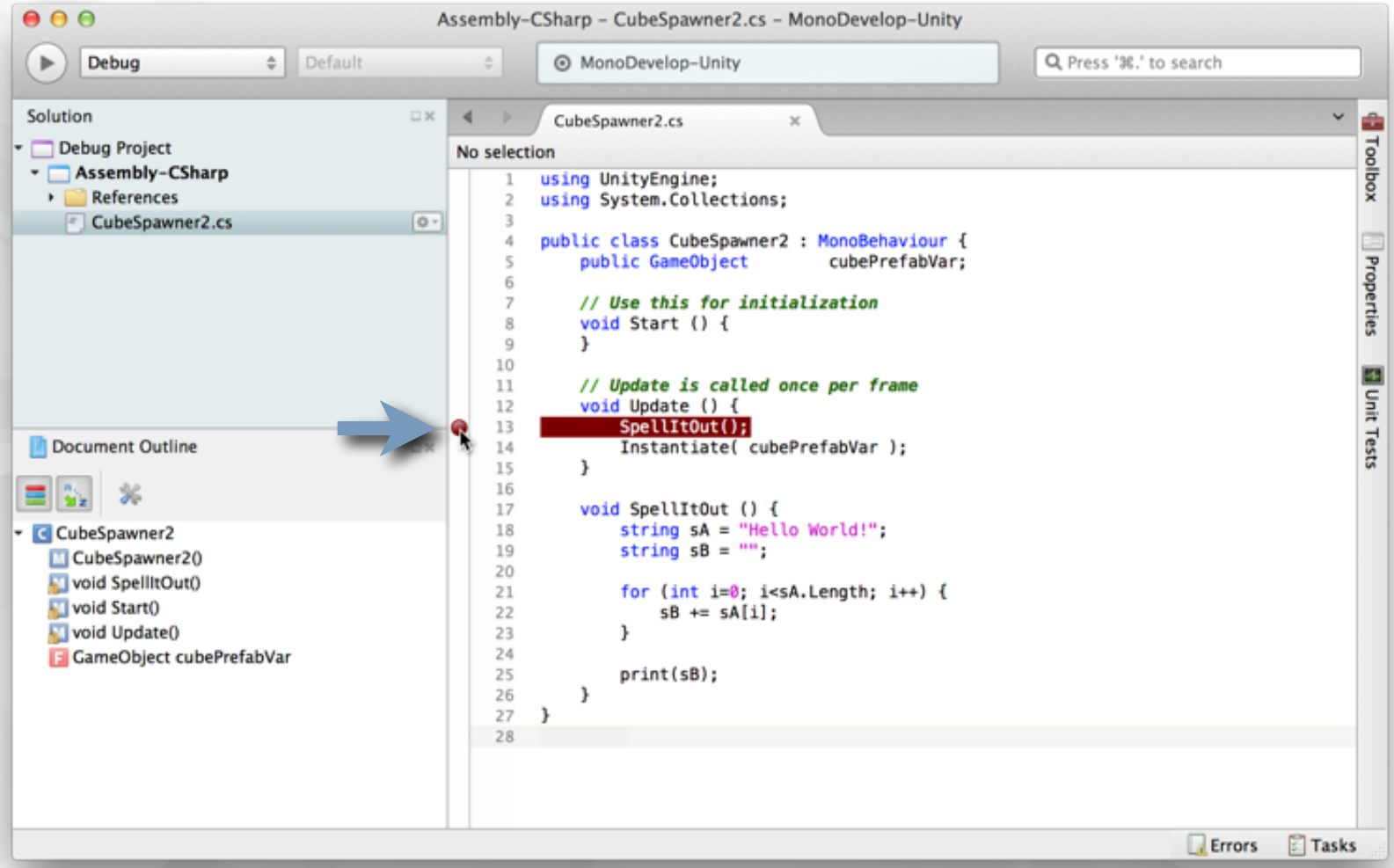
Stepping Through Code with the Debugger

- **Step 1: Set a Breakpoint in your code**



Stepping Through Code with the Debugger

- **Step 1: Set a Breakpoint in your code**



Stepping Through Code with the Debugger

- Step 2: Attach the Debugger to the Unity process

Stepping Through Code with the Debugger

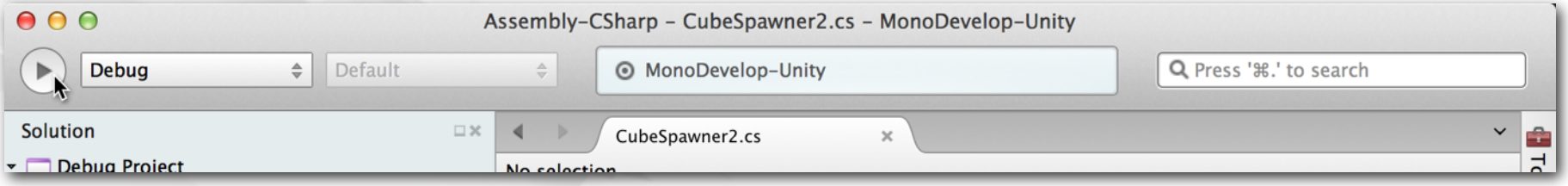
- **Step 2: Attach the Debugger to the Unity process**
  - **Much more detail in the book (about a potential bug)**

Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop

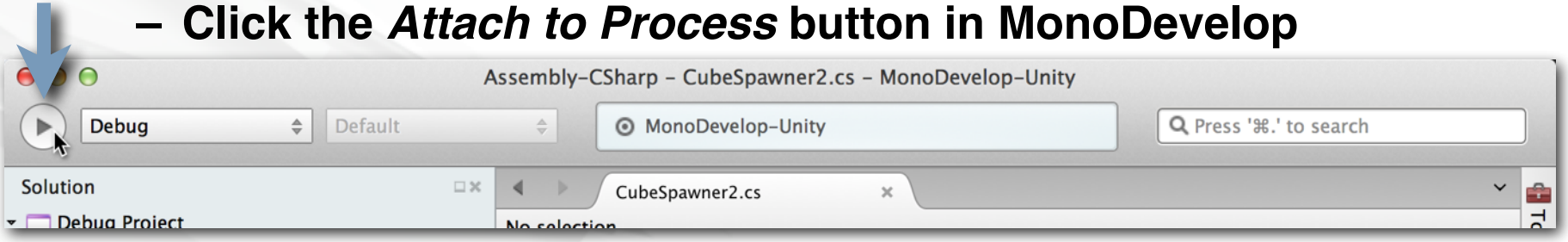
Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop



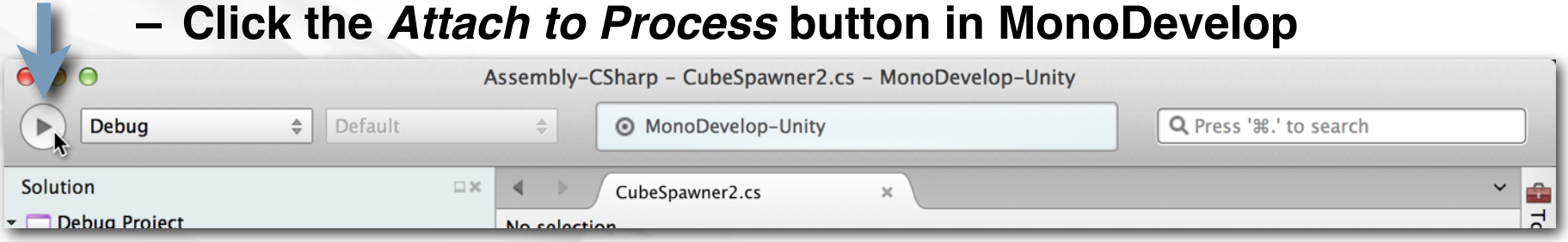
Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop



Stepping Through Code with the Debugger

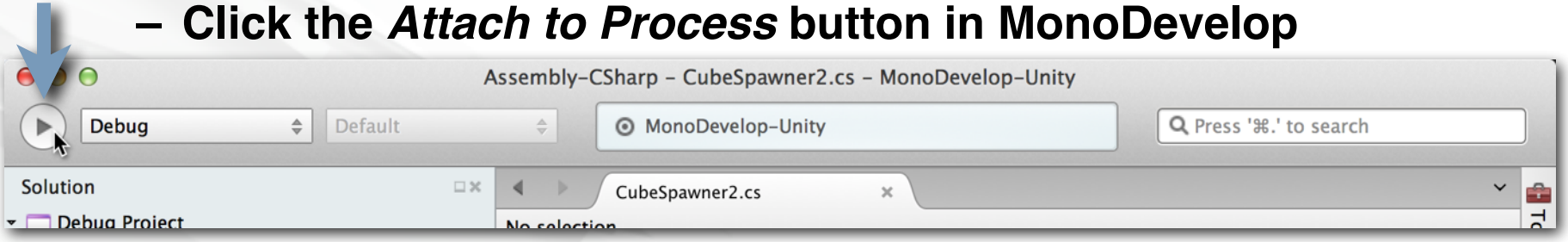
- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop



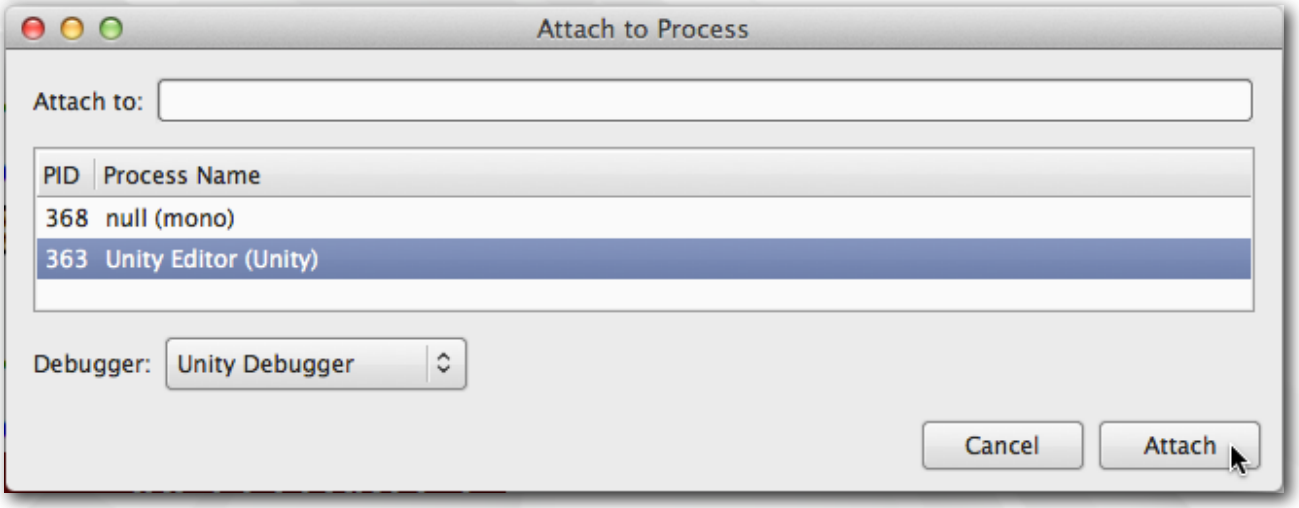
  - Choose *Unity Editor (Unity)* from the process list & click *Attach*

Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop

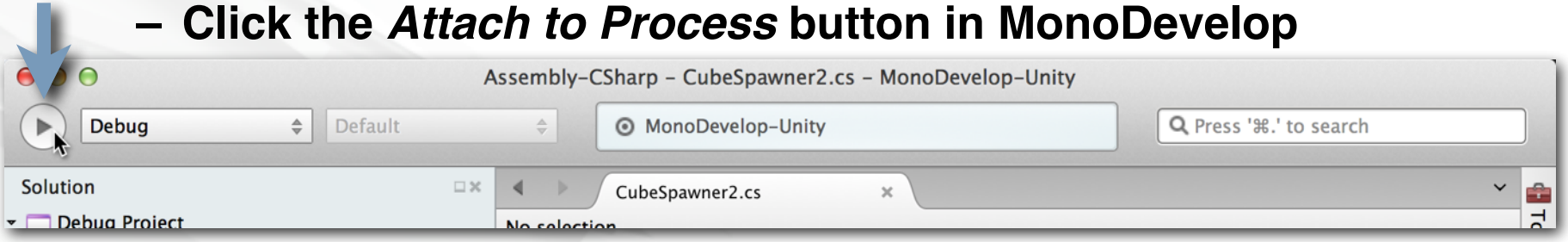


  - Choose *Unity Editor (Unity)* from the process list & click *Attach*

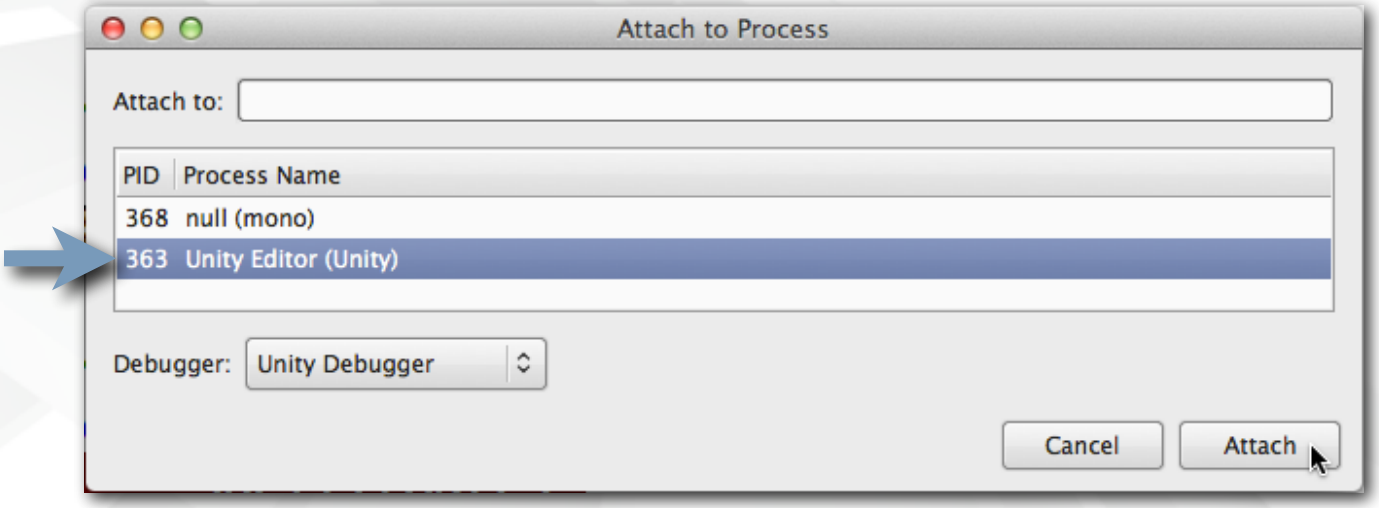


Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop

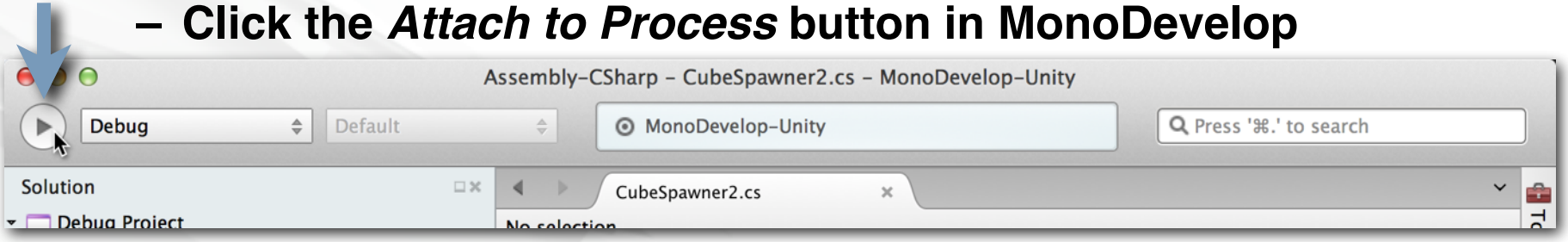


  - Choose *Unity Editor (Unity)* from the process list & click *Attach*

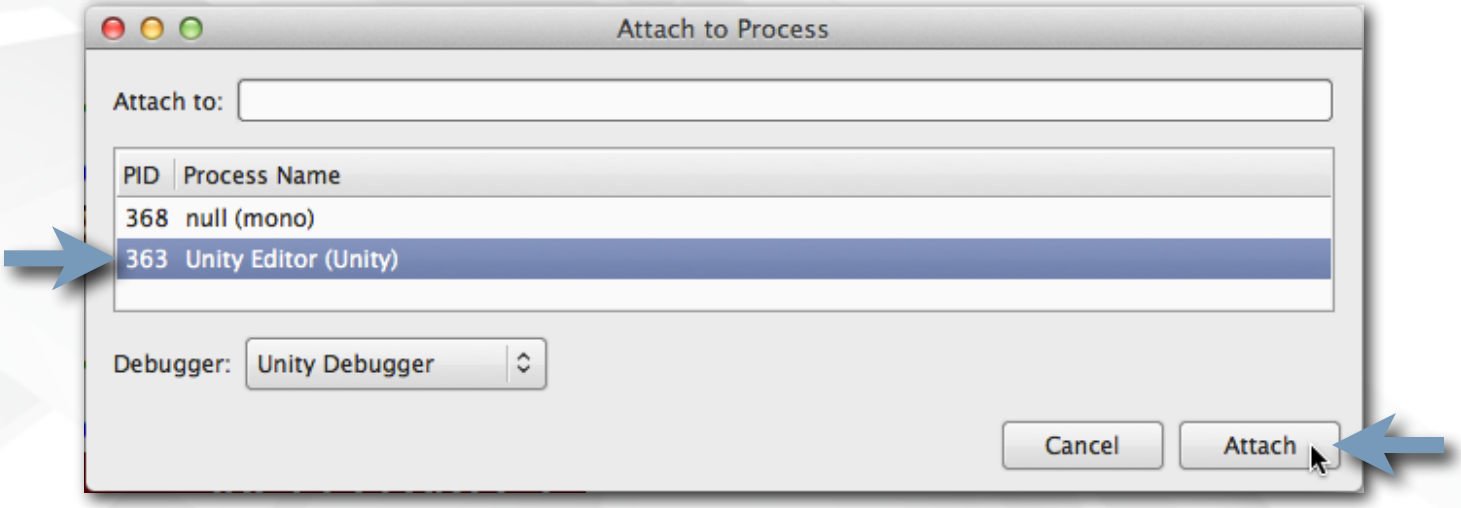


Stepping Through Code with the Debugger

- **Step 2: Attach the Debugger to the Unity process**
  - Much more detail in the book (about a potential bug)
  - Click the *Attach to Process* button in MonoDevelop

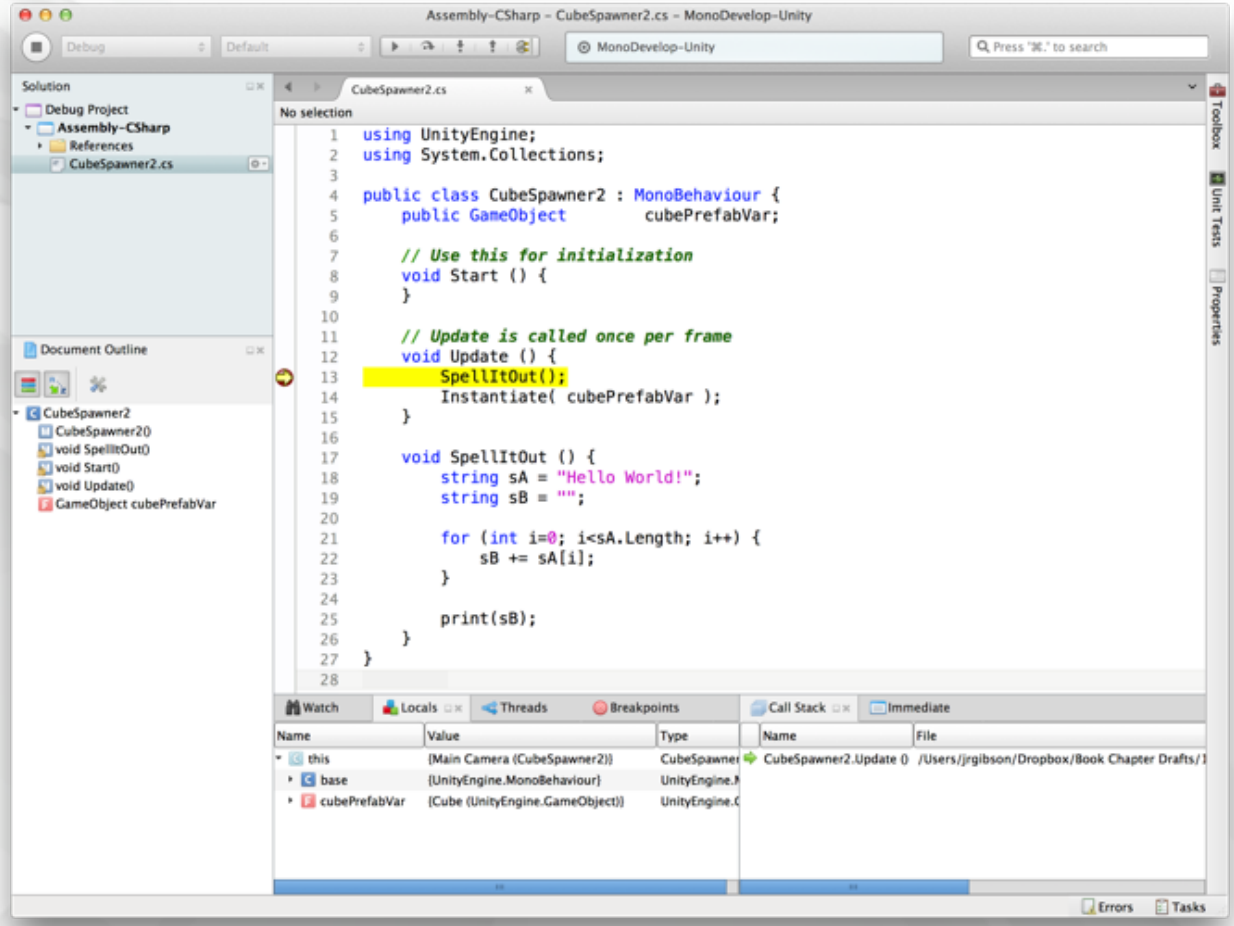


  - Choose *Unity Editor (Unity)* from the process list & click *Attach*



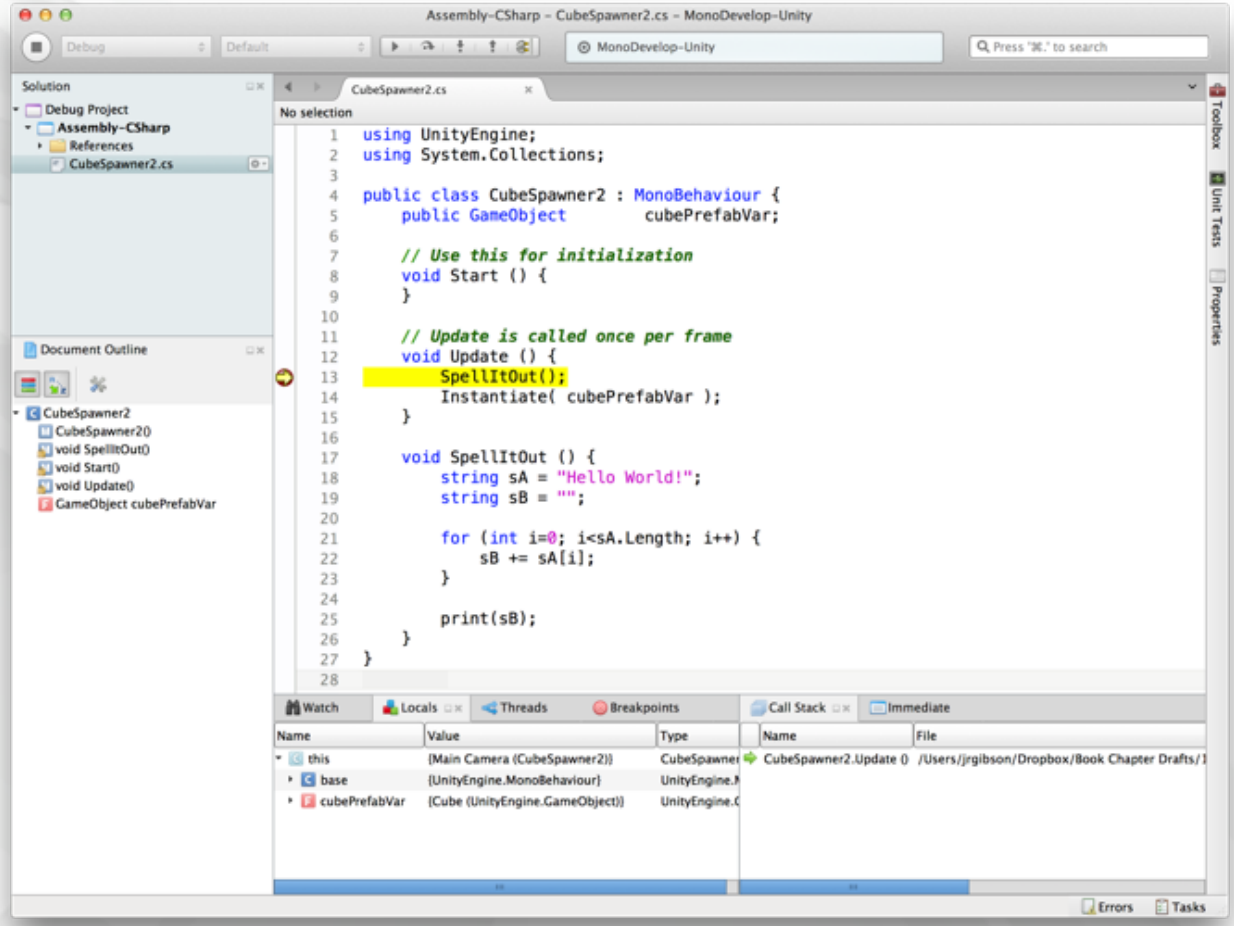
Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**



Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**



Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**

Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**

Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**

Stepping Through Code with the Debugger

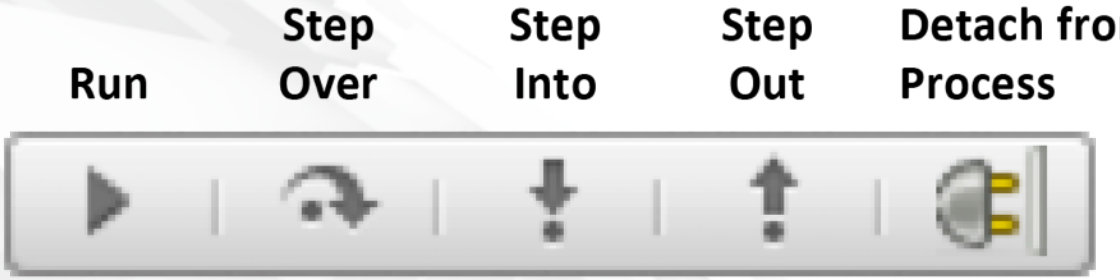
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process

Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**

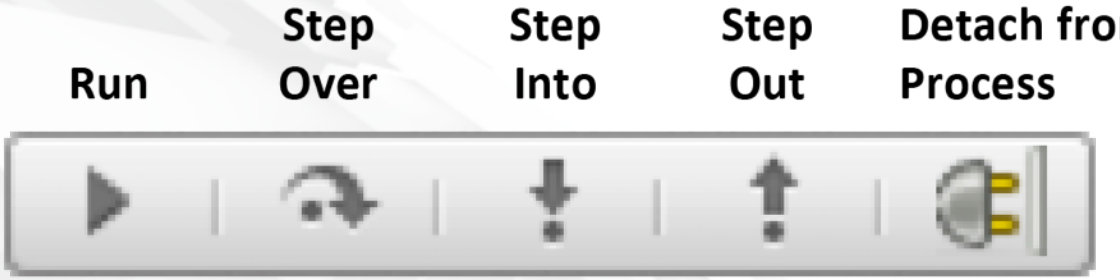
Stepping Through Code with the Debugger

- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



Stepping Through Code with the Debugger

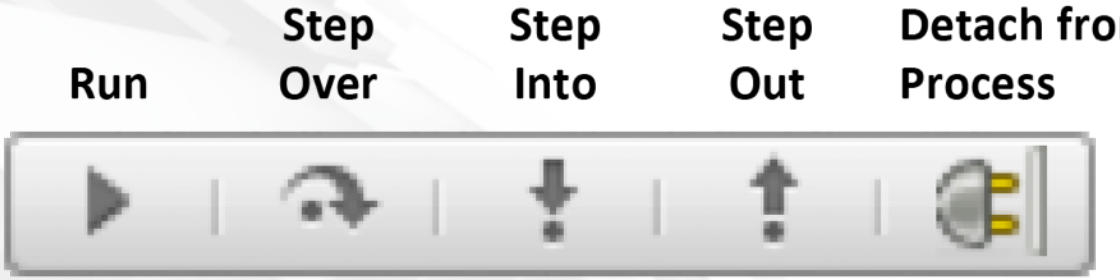
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**

Stepping Through Code with the Debugger

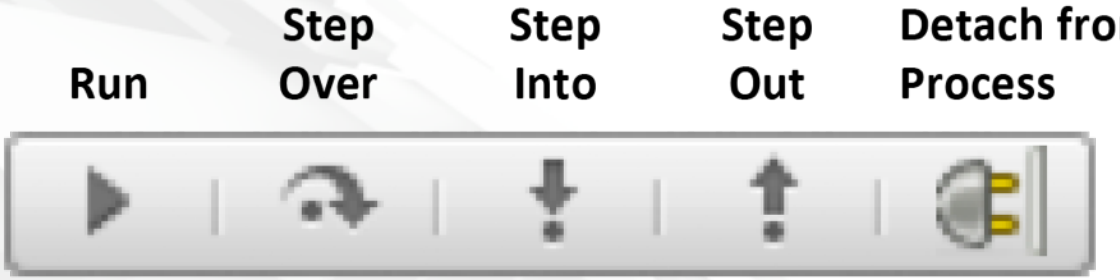
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit

Stepping Through Code with the Debugger

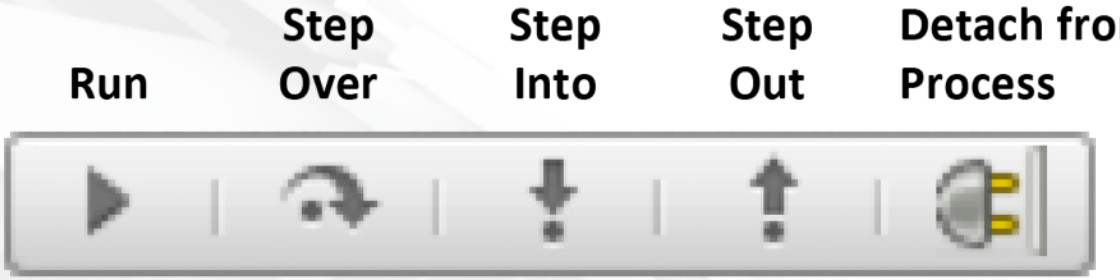
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit
      - If Run doesn't advance to the next frame, switch back to Unity

Stepping Through Code with the Debugger

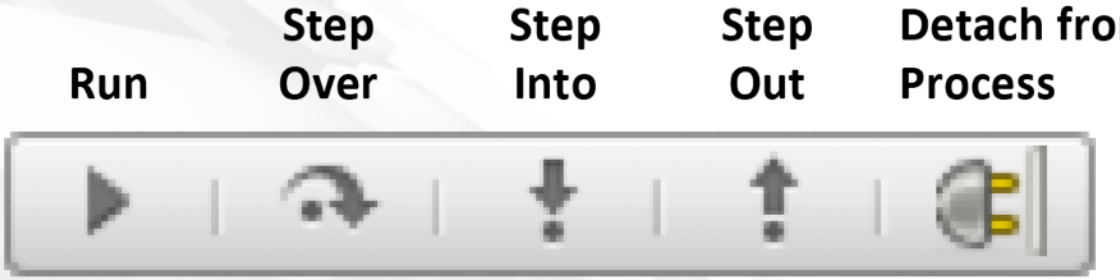
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit
      - If Run doesn't advance to the next frame, switch back to Unity
    - **Step Over** – Continues to the next line, stepping over function calls

Stepping Through Code with the Debugger

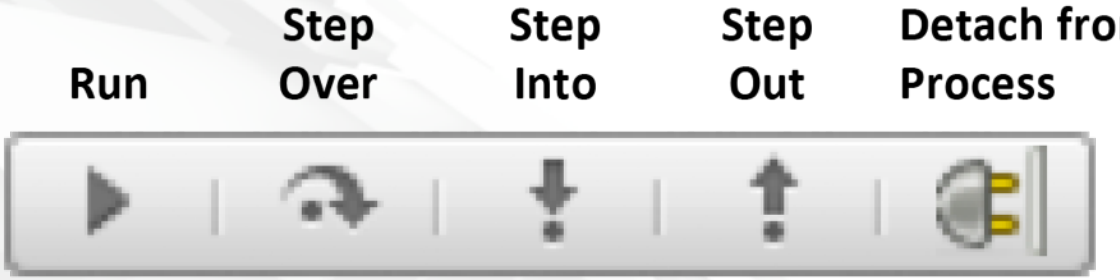
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit
      - If Run doesn't advance to the next frame, switch back to Unity
    - **Step Over** – Continues to the next line, stepping over function calls
    - **Step In** – Continues to the next line, stepping into function calls

Stepping Through Code with the Debugger

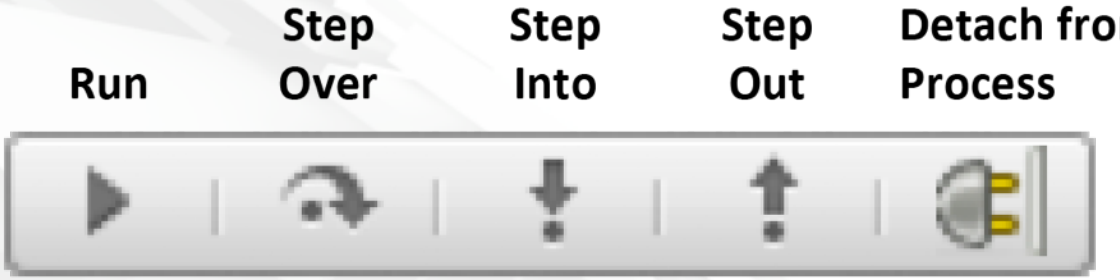
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit
      - If Run doesn't advance to the next frame, switch back to Unity
    - **Step Over** – Continues to the next line, stepping over function calls
    - **Step In** – Continues to the next line, stepping into function calls
    - **Step Out** – Exits the current function but continues debugging

Stepping Through Code with the Debugger

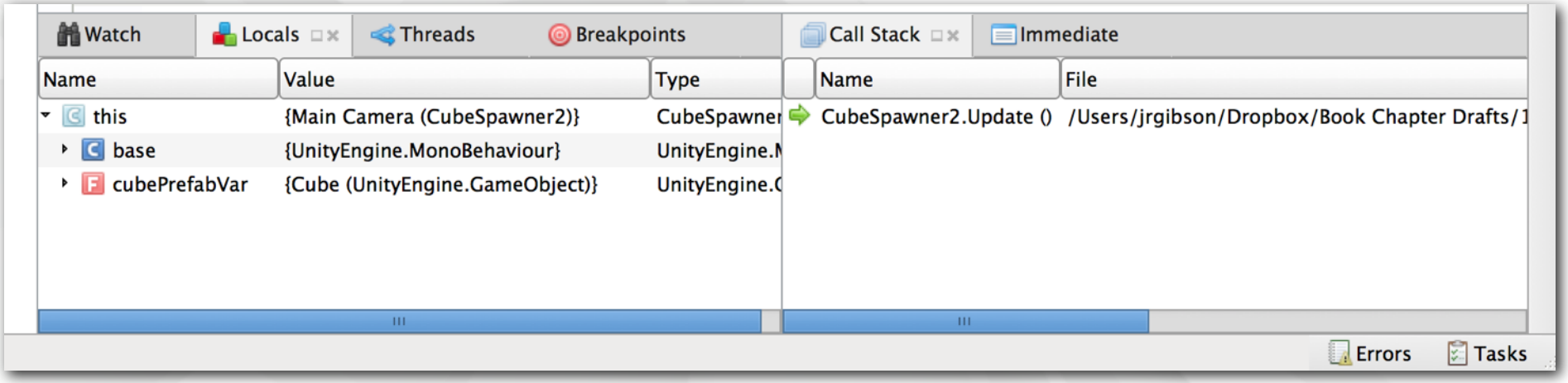
- **Step 3: Click *Play* in Unity**
  - **The Debugger will halt code execution at the Breakpoint**
  - **Unity will be *completely frozen* while the Debugger is halted**
    - This means you **cannot** switch back to the Unity process
  - **Important buttons at the top of the Debugger window**



  - **Each controls the Debugger's execution**
    - **Run** – Continues playing the project until another breakpoint is hit
      - If Run doesn't advance to the next frame, switch back to Unity
    - **Step Over** – Continues to the next line, stepping over function calls
    - **Step In** – Continues to the next line, stepping into function calls
    - **Step Out** – Exits the current function but continues debugging
    - **Detach Process** – Stops debugging altogether

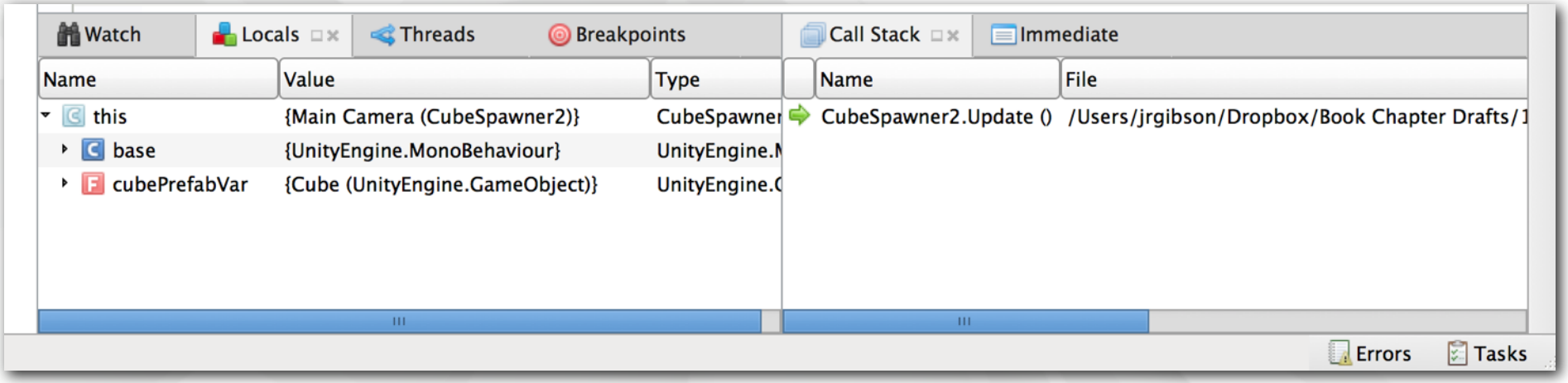
Watching Variables in the Debugger

- Panes at the bottom of MonoDevelop have more info



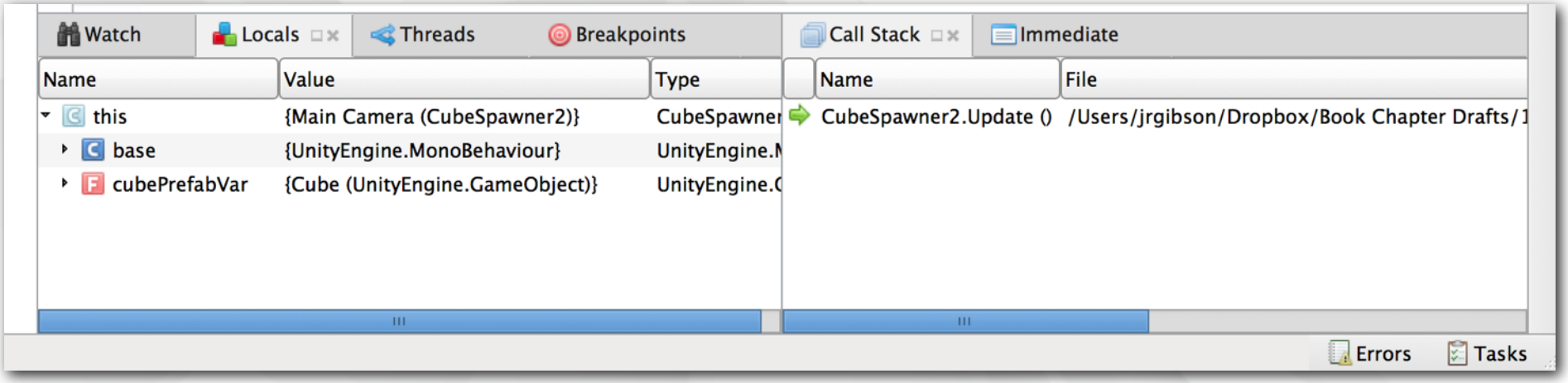
Watching Variables in the Debugger

- Panes at the bottom of MonoDevelop have more info
  - Locals - Allows you to see all local variables



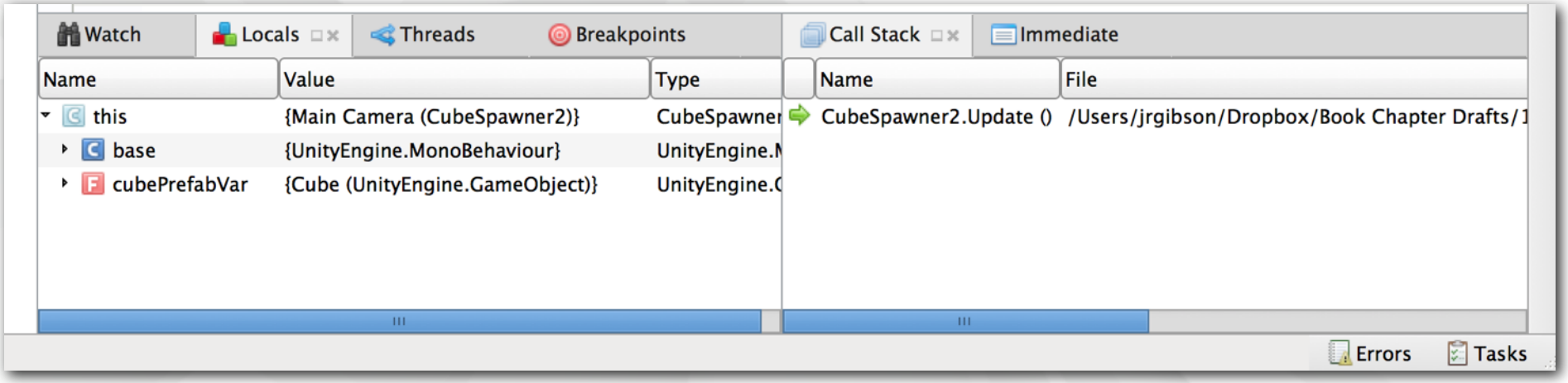
Watching Variables in the Debugger

- **Panes at the bottom of MonoDevelop have more info**
  - **Locals - Allows you to see all local variables**
    - **`this`** is a reference to the current class instance



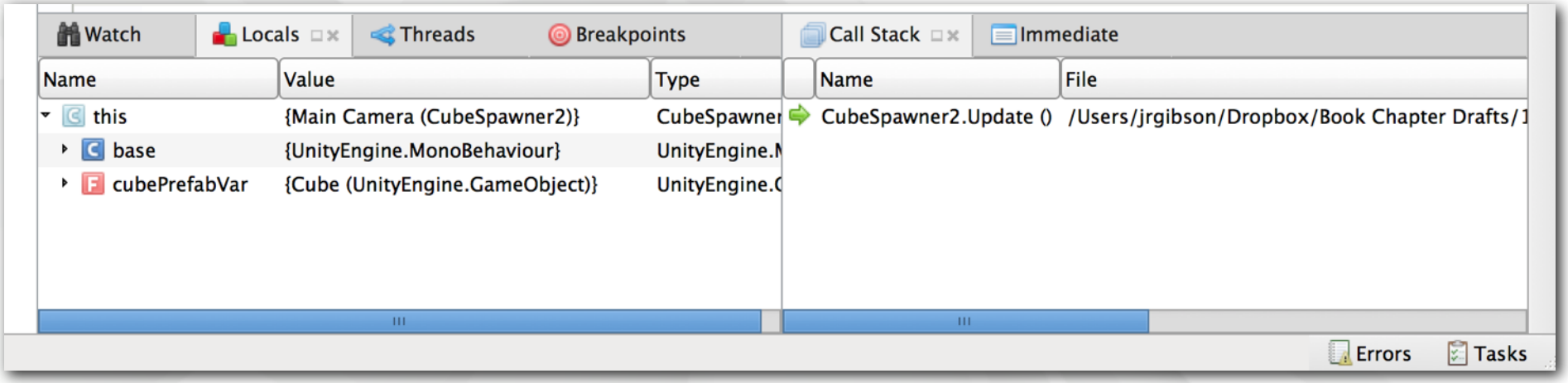
Watching Variables in the Debugger

- **Panes at the bottom of MonoDevelop have more info**
  - **Locals - Allows you to see all local variables**
    - `this` is a reference to the current class instance
  - **Watch - Allows you to enter specific variables to watch**



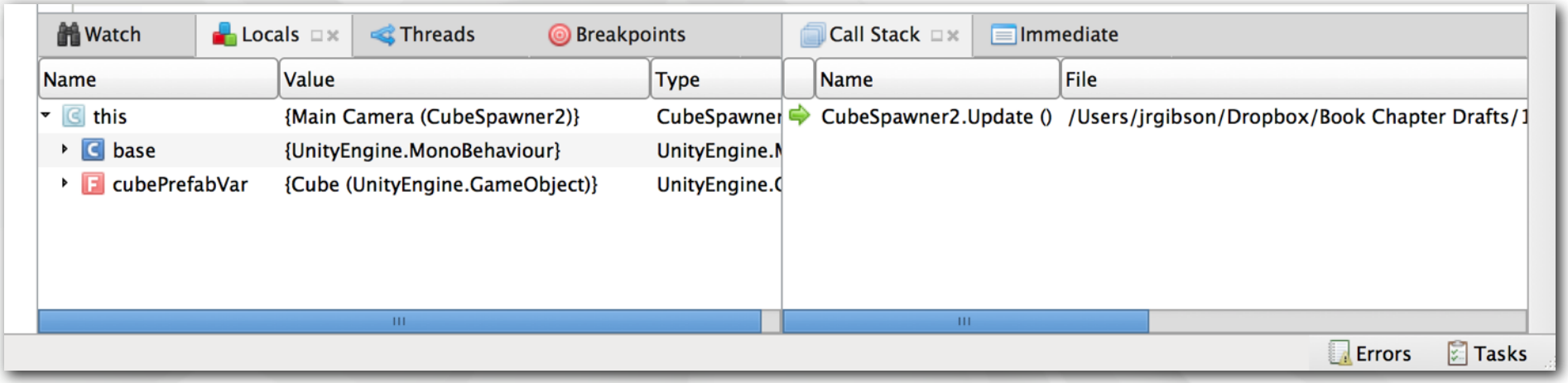
Watching Variables in the Debugger

- **Panes at the bottom of MonoDevelop have more info**
  - **Locals - Allows you to see all local variables**
    - `this` is a reference to the current class instance
  - **Watch - Allows you to enter specific variables to watch**
  - **Call Stack - Shows you which functions have been called to get to this point in the code**



Watching Variables in the Debugger

- **Panes at the bottom of MonoDevelop have more info**
  - **Locals - Allows you to see all local variables**
    - `this` is a reference to the current class instance
  - **Watch - Allows you to enter specific variables to watch**
  - **Call Stack - Shows you which functions have been called to get to this point in the code**
    - Click a function to jump to it's local scope



Chapter 24 – Summary

Chapter 24 – Summary

- Debugging is one of the most important processes in coding

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**
  - **Lots of information online about how to set this up**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**
  - **Lots of information online about how to set this up**
- **Next Chapter: Classes**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**
  - **Lots of information online about how to set this up**
- **Next Chapter: Classes**
  - **Learn about how classes combine data and functionality**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**
  - **Lots of information online about how to set this up**
- **Next Chapter: Classes**
  - **Learn about how classes combine data and functionality**
  - **All the code you write in Unity C# will be in classes**

Chapter 24 – Summary

- **Debugging is one of the most important processes in coding**
  - **The MonoDevelop Debugger is one of the most powerful tools for you to learn**
  - **It's also surprisingly easy to learn to use**
  - **The Debugger can also help you understand complex code**
    - Use it on the code from the book if you're ever confused
- **You can also code for Unity using Microsoft Visual Studio, which has its own debugger**
  - **Lots of information online about how to set this up**
- **Next Chapter: Classes**
  - **Learn about how classes combine data and functionality**
  - **All the code you write in Unity C# will be in classes**
  - **Classes are also the key to Object-Oriented Programming**