
AFTER-SCHOOL
PROGRAMMING

WELCOME TO YOUR AFTER-SCHOOL PROGRAMMING CLUB!

Programming is a fun and rewarding way to learn about the technology that surrounds us, and anyone can learn it. Kids love building their own applications and web pages.

Codecademy is a fun and social way to learn coding **for free**. The learning takes place through our interactive lessons and projects. Our installation-free tool combines instruction and programming in one interface, which frees you up to work with students one-on-one.

This kit contains everything you need to start your own after-school programming club, even if you don't have **a background in computer science**. Please let us know if you have any questions or feedback—we would love to hear from you!

-**Sasha Laundy**, Codecademy teacher in residence

contact@codecademy.com

TABLE OF CONTENTS

WELCOME

Why teach programming?

Kids spend an increasing portion of their lives interacting with and through screens about which they know little or nothing.

The more they do so, the more they accept the values of Facebook, Google, or iTunes as pre-existing conditions of the universe. Instead of opening their minds, technology shuts them down. Compounding all this, the few places most young people have available to learn about computers tend to teach them how to use and conform to existing software applications rather than how to make their own.

Introducing kids to code reveals to them how computers are really “anything” machines, capable of doing pretty much anything we program into them. It gives them the ability both to read and to write in the foundational languages of the digital age and, in doing so, fundamentally transforms their perspective from that of user to maker, consumer to creative.

-**Douglas Rushkoff**, author, Program or be Programmed

rushkoff@codecademy.com

GETTING SET UP

Codecademy works best as a self-guided tool. Since the instruction and the programming are in the same interface, you are freed up to work with students who need extra help or to help contextualize and extend the standard curriculum.

Try to get the students comfortable making mistakes and trying new things. They will learn a lot from each mistake that they make. Show them basic error-finding techniques and how to research the answers to their questions on the Internet.

You may also find that students can answer their own questions much more effectively if they are working in pairs. Professional programmers use these techniques—debugging and pair programming—every day.

Begin a class by telling the students what they are working towards. Most courses follow a model where the exercises introduce the topic and the course ends with a practical project. For each lesson we've also suggested some talking points.

Since our lessons are self-guided, they are also self-paced. Some students might zoom through the exercises quickly, and others may take more time. We have suggested sources of extra help and activities that more advanced students can tackle elsewhere in this guide.

CHECKLIST TO GET STARTED

- [ ] **Get to know Codecademy.** Set up a Codecademy account for yourself. Go through the exercises and get to know the curriculum.
- [ ] **Size & age.** Decide how big you want the group to be and what age range you are targeting.
- [ ] **Scope.** What kind of program do you want to start? Is it homework help, science club, or a teaser for AP Computer Science? A web programming elective?
- [ ] **Time.** Pick a weekly gathering time.
- [ ] **Resources.** Will you need funding for your program? How will you staff your program? Are there local programmers, college, or high school students who could help? High school students with previous experience can be teaching assistants.
- [ ] **Location and equipment.** If you don't have a computer lab, consider libraries, learning centers, or youth media centers.
- [ ] **Promote.** We've included a sample poster you can copy and hang in your school. If you have a core group of students who are excited about the club, they may be the best way to advertise.
- [ ] **Parental approval.** give your parents some background information. Maybe they will be interested in taking the lessons with their kids!
- [ ] **Share.** Your lessons learned and best practices will help other teachers and students. If you have a cool story to share, tell it in the forums or let us know. And if you need any help, please e-mail us contact@codecademy.com.

FAQS

If I don't know how to program, can I use this with the kids?

Yes! We give you everything you need to get started. We recommend that you familiarize yourself with the material ahead of the students, but you don't need to be a professional programmer to teach. If students are stuck, you can model debugging and researching problems—two very important skills for any programmer.

There are lots of other resources if you get stuck. Other teachers are communicating on a custom Codecademy forum. Sites like Stack Overflow offer answers to almost every problem you might encounter. Students, parents, and professional programmers in your area may be interested in volunteering their expertise. And we've included resources in this packet for further reading and help.

What kind of preparation do I need to implement the program?

You'll need to work with your school to set up a meeting time, arrange for access to Internet-enabled computers, supervise the session, and we provide the rest! Everything you need is in this kit. We provide you with a flexible session plan, so you can get started right away.

Is Codecademy free to use for schools?

Yes, creating and taking lessons on Codecademy is completely free.

What kind of computers or browsers do I need?

There is nothing to install, so you can get started right away by going to our website at http://codecademy.com. Any kind of laptop or desktop computer will work—Mac, Windows, or Linux. We strongly recommend using Chrome or Firefox as a web browser. You can also use Internet Explorer 9. Our site does not support iPads, Internet Explorer 6-8 or Opera.

What do students need to get started?

They will need an e-mail address in order to create a Codecademy account.

What will students learn during these sessions?

After the lessons students will be able to create their own websites through hands-on exercises. They'll also be familiar with the basics of programming in the JavaScript language, a modern and flexible language with wide applications in web programming. They will be well prepared for learning any other programming language, including Java for AP or college courses.

What should I do with kids who finish early, or students with prior experience?

This is a great leadership opportunity for those students. Ask them to help beginners or share projects they've made. They may be ready for our more advanced jQuery track at http://codecademy.com/learn. Encourage them to pick a small project and research how to do it—this is a skill all programmers need. We've included even more resources at the end of this packet for students who have finished our entire curriculum and are eager for more.

How do I tailor the program to fit my schedule?

We've included a suggested schedule as a starting point in this packet. Each lesson is designed for an hour-long session. Since the lessons are self-paced, not all students have to move through the material together, so you have considerable flexibility in how you structure your time. We'd love to hear about what works and what doesn't for your students.

Can I measure my students' progress?

You can follow your students' progress by following their point count and badges, which are displayed on their profiles.

What if kids aren't ready for programming, or struggle with early lessons?

We encourage you to try these lessons with your students. We've seen great success with learners ages 7 to 87. If students are struggling or are finding it too abstract, there are several excellent programs that provide a more visual interface: Alice, Scratch, Mozilla Thimble, Hackasaurus, Blockly. They can provide a good warmup to JavaScript or Python programming.

Who designs your courses?

The courses are created and curated by programming educators in the community. The lessons go through several rounds of testing prior to their release. We at Codecademy make sure that all our courses are of great quality.

Can I make my own courses for my students?

Yes! We offer a free tool that lets you create lessons just like our core curriculum, and you can share them with students all over the world. Go to codecademy.com/teach to create your own courses.

JAVASCRIPT CURRICULUM GUIDE

Intro

This is a 14 week curriculum that covers HTML, CSS and the basics of JavaScript. During the two semesters, students will build several websites with HTML and style them with CSS. The sessions will end with making a small game in JavaScript, a versatile web programming language. Most courses follow a model where the exercises introduce the topic and the course ends with a practical project. For each lesson we've also suggested some talking points.

You're not confined by the age or ability level of the student, since the class model is very flexible. Students can work through the exercises at their own pace. We've given you a suggested starting point with this schedule. Adapt liberally for your students—focus only on web pages for a shorter program, jump right into JavaScript if it's programming you're after, and if you have more advanced students we have suggested further study at the end of this packet.

Programming requires experimentation, research, and a little bit of fiddling. Empower students to play with things to find out how they work. Help them to teach themselves, and each other, by creating what they want to create, and allowing the students input in regard to the direction of the class.

WEEK ONE: INTRODUCTION TO CODING

http://www.codecademy.com/tracks/afterschool-semester1

Go through what's going to happen during the semester: everyone will be coding websites and games. Introduce Codecademy and make sure everyone has an e-mail address and can register to the site. Emphasize that everyone learns at a different pace, experimentation is welcome, you will learn lots from catching and fixing your mistakes. This is also different from a normal class because you get to build real things while you learn.

Ask students what their goals are, what they already know about technology and the Internet, and what questions they want to get answered by participating in your club. If you have time, let students try the first exercises on the track.

Discussion:

- What is programming?
- What is a browser?
- What is HTML (structure), CSS (presentation), Javascript (behaviour/interaction)
- Where can you see HTML & CSS? View the source code of your school website.
- Where in the everyday world one can see programming?
- What knitting and mathematics have to do with programming?
- What would happen if computers disappeared?
- What will happen in the future as computers get smaller, faster, and cheaper?
- Look up the history of the printing press and how cheap paper manufacturing radically changed access to reading and writing across economic classes. Can you see any parallels to programming and our modern society?

WEEK TWO: HTML FUNDAMENTALS

http://www.codecademy.com/tracks/afterschool-semester1

HTML is the bones under every single web page on the web. You will learn the fundamentals of HTML to make your own basic website. You will include images, organize text, and add links to your page.

Have students start with the first lesson, which covers the following topics:
Structure of HTML - Basic tags - Hyperlinks - Images
Then they can go on to the first **Project:** Build Your First Webpage.

Discussion:

- Look at the source of a favorite website and see if you can find any <p> elements (use Ctrl-F). What other familiar tags can you spot?
- Show students how to use shortcuts to copy-paste code. For Windows it's ctrl + c and ctrl + v, for Mac cmd + c and cmd + v.
- Note that writing HTML and CSS is not technically "programming", but formating. HTML and CSS are called markup languages that the students need to know in order to create a canvas for web applications and programs that you will write in JavaScript.

WEEK THREE: MORE WITH HTML

http://www.codecademy.com/tracks/afterschool-semester1

Learn more about the fundamentals of HTML. You will further organize the material on a webpage using lists and tables, which will be extremely useful down the road. You also learn how to bold, align, and add color to your text!

Topics covered in the lesson: Lists - Styles - Tables

Project: Make a Recipe Card

Discussion: Ask everyone to look at http://w3schools.com/html and choose a tag from the left-hand column to research and present in the next session.

WEEK FOUR: HTML REVISION

http://www.codecademy.com/tracks/afterschool-semester1

Go through what you have learned so far and have students present what kind of sites they have built. Now is a good time to make sure they understand that indentation is very important, both so others can read their code and so they can make sure to catch any errors or unclosed tags.

Make sure students remember at least the following:

<body>.

<p> tags and <h1> .. <h6> headers.

, .

<img src= ”” >

WEEK FIVE: STYLE WEBPAGES WITH CSS

http://www.codecademy.com/tracks/afterschool-semester1

If HTML is the bones of a webpage, CSS is the skin. It is a powerful tool that lets you style simple pages or complex websites. Integrate HTML and CSS to make a beautifully formatted page.

Topics covered: CSS - Selectors - Different ways to link CSS

Project: First Website Using HTML and CSS

Discussion:

- Learning CSS is a lot of fun: encourage students to experiment with colors, fonts and overall styling to make their pages ugly or beautiful.
- Teach them how to find CSS code, either with Firebug or by saving the website on their computer.
- Send them to CSS Zen Garden, a collection of pages that share exactly the same HTML. Only the CSS is different, but the pages look nothing alike. http://www.csszengarden.com/

WEEK SIX: ADVANCED CSS SELECTORS

http://www.codecademy.com/tracks/afterschool-semester1

You've seen a glimpse of the magic of CSS selectors. Now it's time to grasp the full power and make the Internet pretty once more!

Topics covered: Using selectors and IDs - Inheritance and Cascading - Pseudoselectors

Project: Build a Resume

Discussion: As they build their resume, have the students imagine they're applying for a real job when they are 22.

- If they are younger, have them invent their own personalities with fun and weird backgrounds.
- High school seniors could create a college application website with information about their work and activities.

WEEK SEVEN: INTRO TO CSS POSITIONING

http://www.codecademy.com/tracks/afterschool-semester1

Learn the basics of positioning elements on the page using CSS. Start with the box model. Then float and clear your way to lovely websites. You will learn the box model, which is fundamental to understanding positioning. Be patient as you learn it—it may require some practice to thoroughly learn it. Diagramming your page on paper before you build it can be extremely helpful.

Topics covered: Box model - relative positioning - absolute positioning

Project: Create a Personal Webpage

Discussion:

- Have a short look into the history of computer science: Who is Alan Turing? Ada Lovelace? Tim Berners-Lee? Steve Jobs? Linus Torvalds? James Gosling? Grace Hopper?

WEEK EIGHT: ADVANCED CSS POSITIONING
http://www.codecademy.com/tracks/afterschool-semester1

Build on the previous lesson by exploring absolute and fixed positioning, which are important tools for building any web page. Includes a review of relative and static positioning.

Topics covered: Absolute positioning - fixed positioning

Project: Pizza Time!

Discussion:

- Teach the students how to use Firebug if they are using Firefox. It's a tool that lets you inspect the structure of web pages.
- Adding a style that gives every div a border of 1px is ugly but can help you see what's going on.
- If you're using the Chrome browser, you can right-click and choose "Inspect Element." You will see the page's styles on the right. You can edit the styles right in the browser and see it change live on the page.
- Ask everyone to look at http://w3schools.com/css and choose an attribute to present in the next session.

WEEK NINE: CSS REVISION

http://www.codecademy.com/tracks/afterschool-semester1

Go through what you have learned so far and have students present what kind of sites they have built. Now is a good time to make sure students know how to format CSS stylesheets properly, indenting blocks with two spaces and leaving blank lines in between blocks.

Make sure students remember at least the following:

selector { property : value;}, color, background-color, where to put your CSS and how to link to it from the HTML page

New activity: save your HTML to a file in a text editor that allows you to save the file as plain text (important—saving it as a **Microsoft Word or RTF file will not work)** with the extension .html. Do the same for the CSS. Then open your HTML file in a browser to view your web page!

WEEK TEN: GETTING STARTED WITH PROGRAMMING
http://www.codecademy.com/tracks/afterschool-semester2

An introduction to JavaScript, a friendly programming language. Note that this is their first exposure to programming, which is a completely different paradigm from HTML. Refer students to need to think slightly differently, and be patient as they learn a completely new skill. This lesson will introduce them to some fundamental building blocks, which will be revisited later. Programming is fun and rewarding once you get the hang of it!

Topics covered: Strings - Numbers - Variables - If/Else statements
Project: Make your own adventure game!
Discussion: Make your own flashcards: Collect a list of new terms from every session and make flashcards for students to practice with. You can find help from codecademy.com/glossary. Can you think of any good programming jokes?

WEEK ELEVEN: INTRODUCTION TO FUNCTIONS
http://www.codecademy.com/tracks/afterschool-semester2

Functions store blocks of code that can be called upon any time to avoid repetition. They are one of the most fundamental concepts in all of computer programming, but can be tricky at first.

Topics covered: Functions - Using variables in functions - Returning a value at the end of a function.

Project: Rock, Paper, Scissors

Discussion:

- Remember to review last week's topics: strings, numbers, variables and if/else statements in the beginning of the class.
- Why are functions so useful to programmers?
- What are some useful analogies that can help you understand and explain functions?

WEEK TWELVE: INTRODUCTION TO FOR LOOPS IN JS
http://www.codecademy.com/tracks/afterschool-semester2

Loops are a series of instructions that repeat until a condition is met. This is extremely useful for automating repetitive tasks.

Topics covered: For loops (one type of loop)

Discussion:

- Why are loops so useful?
- What else could you do with a for loop?
- If students are having trouble with this concept, consider acting it out.
- Stepping through a program line by line and seeing how the variables change can be useful for helping students understand how the programs work.

WEEK THIRTEEN: RECAP OF JAVASCRIPT

http://www.codecademy.com/tracks/afterschool-semester2

Go through what you have learned so far and have students present what kind of games they have built. Make sure students remember at least the following:

Topics covered:

Javascript Revision

var myName = “Ryan”;

Output: console.log(), alert(), confirm()

Operators: +*/-%

For loops: for(i=0;i<5;i++) { }

Comparison operators: >, <, >=, <=, ===, !==

Branching: if() { } else if() { } else { }

Tip: Ask a programming-savvy parent or volunteer to visit your classroom and explain what they do in their work. You can also reach out to local software companies their programmers are often willing to share their experience.

WEEK FOURTEEN: PUTTING IT ALL TOGETHER

http://www.codecademy.com/tracks/afterschool-semester1

Celebrate what you've learned. Have students complete their own personalized projects.

Project: Code'n'Tell

DEBUGGING YOUR CODE

Finding and fixing mistakes in code is an important skill for any programmer.

General tips

- Error messages can provide clues. If you can't decipher them, you can often Google them.
- A frequent source of mistakes for beginners is syntax—a wayward comma or a missing bracket. Software programs are very picky about every single character, so check them carefully. We run a program called a linter that checks your code for syntax errors, and will display either a red or yellow icon to the left of the problem line. If you hover over the icon, it will tell you with the problem is.
- Logic problems: trace through the program line by line. Speaking the program out loud can really help you spot errors more quickly. Programmers call this rubber duck debugging (http://en.wikipedia.org/wiki/Rubber_duck_debugging).
- Printing out values of your variables at various places in the program can help you reveal this sort of error. Once you get to be a more advanced programmer there are more sophisticated debugging techniques, but this one will really help with simpler programs.

HTML & CSS

- Did you include <body>,<head> and <html> tags?
- Make sure to link the correct CSS stylesheet so the HTML page can find your styles.
- Make sure you remember to close the tags in the right order. They should nest, so the most recent one opened should be the first one closed: {()} rather than {()}.
- Make sure your image titles have the correct name and extension (.jpg, .png).
- Classes and IDs are easy to confuse in CSS. Use a period for classes and a pound sign (#) for IDs.

JavaScript

- Variable names are case-sensitive and even changing one character will refer to a different variable.
- Make sure you used the right sorts of brackets (there are several: (), [], and {}) and that each open bracket is closed.
- Did you include enough semicolons? Are there too many semicolons? If you wrote a function but it's not running, did you call the function?

TROUBLESHOOTING

For fixing your code, see the Debugging section of this document, and for instructional help, see the teacher forum on our website.

Install the most up-to-date version of Firefox or Chrome. Codecademy works most reliably on these two browsers. Internet Explorer versions 6-8 are not supported—please use version 9. iPads are not currently supported but any kind of laptop or desktop computer should work fine.

Check that there are no firewalls that block Codecademy, and that cookies are enabled.

Clear your browser cache and restart the browser. You can clear the browser cache in your browser settings or preferences.

Check that all of your plugins are up-to-date. You don't need to have flash installed to use Codecademy. You can check the status of your plugins for Firefox, Chrome, and Safari here: http://www.mozilla.org/en-US/plugincheck/

For most common issues, see: http://help.codecademy.com/ You can also reach out to us via contact@codecademy.com.

I'M DONE! WHAT'S NEXT?

More resources, links, and tools for students who are ready to move on.

Learning Python and jQuery

Python. For students who want to learn a great all-purpose programming language that can be used for a very wide range of applications in various disciplines, check out the Codecademy Python track.

jQuery. For students who are passionate about web programming, check out the Codecademy jQuery track to take your JavaScript to the next level.

Designing web applications

Have students imagine applications or websites they'd like to use every day, and then research how to build them.

Students can read about the basics of web design that some of these sites:
Smashing Magazine http://www.smashingmagazine.com/
A List Apart http://www.alistapart.com/topics/design/

While thinking about the layout of a page, simple mockup tools can be very helpful:
Balsamiq http://balsamiq.com
Mockingbird http://gomockingbird.com

You can read more about color sets for websites here:

http://colourlovers.com

http://kuler.adobe.com

And there is more information about fonts here:

http://www.google.com/webfonts/

Challenges

Mathematically inclined students can tackle the following mathematical challenges in any programming language they choose Project Euler:

http://projecteuler.net/

Or they can test their code-fu by making programs as short as possible with Code Golf:

http://codegolf.com/

Tools

Version control is an extremely important tool for programmers. Pro Git (http://git-scm.com/book) offers a great introduction to using Git for version control. With Git, programmers can share their code and collaborate with others around the world on Github:

http://github.com

PRINTABLE RESOURCES

LETTER TO PARENTS

Most parents will be okay with their kids picking up new skills, but some might need more information. Send this letter home with your students and everything will be crystal clear.

LETTER TO PRINCIPAL

If you need some help convincing the rest of the staff, we've written a letter to your principal for you.

POSTERS!

Post these up around your school to inform kids that they could be creating their own games and apps with their friends.

DEAR PARENTS,

Your child will be participating in our after-school computer programming club this semester. Programming is part of a broader body of digital literacy skills that are increasingly important as more and more of our life is managed by digital devices.

Students will learn about how the Internet works and how software is made by building custom applications, games, and websites. Since the curriculum is self-paced, they will also learn how to troubleshoot and teach themselves new skills.

We will be using interactive programming lessons from Codecademy (http://codecademy.com). In order to participate, your student will need an e-mail address to make a Codecademy account. You can get free email accounts from your Internet service provider or free email providers like Gmail, Hotmail, or Yahoo! mail.

We hope you enjoy seeing the software that your child builds and discussing technology topics with them. Please be in touch if you have any questions or feedback on the program.

Sincerely,

DEAR PRINCIPAL,

A teacher in your school has taken the initiative to plan an after-school computer programming club this semester. The program will run for 14 weeks with one hour weekly sessions.

- Programming is part of a broader body of digital literacy skills that are increasingly important as more and more of our life is managed by digital devices.
- By learning basic programming skills, more and more people can have a meaningful role in creating the next generation of technological advances.
- Programmers are in extremely high demand in the job market, and demand is rapidly outpacing supply.
- Even for professionals who are not programmers, with basic programming knowledge they can be more effective at their jobs and do tasks and analysis they could not otherwise accomplish.
- Learning to program at a young age makes future study much easier. Many colleges expect students to come in already having programming experience.

The students will be using free, interactive programming lessons and projects from Codecademy (http://codecademy.com). There is nothing to install—the tool is run in the browser. And since the curriculum is self-paced, they will also learn how to troubleshoot and teach themselves new skills that professional programmers use every day.

Through these lessons and projects, students will learn how the Internet works and how software is made. They will learn this not by reading, but by building their own programs and webpages from scratch. They will join a new generation of makers, not just consumers, of software, and will help shape our digital future.

Sincerely,

Zach Sims, CEO & co-founder Codecademy

After School Coding

Build Awesome
Games & Apps
With Friends!

When:
Where:
Bring:

CODING CHEAT SHEET // CSS

COMMENTS

```
/* This is a multi-line
    CSS comment. */
```

COLORS

```
#RRGGBB (long)
#RGB    (short)
00 is the lowest, FF is the highest
You can also use the names in word form:
‘color:red’
```

SELECTING TAGS

```
h1 {
  color: red;
}
```

SELECTING CLASSES

```
.nav {
  background: #333;
}
```

SELECTING IDS

```
#top {
  background: #f00;
}
```

SELECTING CHILDREN

```
div > img { ... }
```

PSEUDOCLASSES

```
:hover, :link, :visited, :active, and
:focus.
a:hover {
  color: #000;
}
```

MARGIN AROUND A DIV

```
#box {
  //    top right bottom left
  margin: 10px 5px 10px 5px;
}
```

PADDING WITHIN A DIV

```
#box {
  // 10px on all sides
  padding: 10px;
}
```

CODING CHEAT SHEET // HTML

COMMENTS

```
<!-- comment -->
```

OPEN AND CLOSE TAGS

```
<tag attribute='value'>content</tag>
```

SELF-CLOSING TAGS

```
<br />
<img />
```

SKELETON HTML PAGE

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
  ...
  </body>
</html>
```

LINKS

```
The following text is <a href= "someurl.com"
>hyperlinked</a>
```

DIVS

```
<div id="myID" class="myClass">...</div>
```

HEADERS

```
<h1> This is a header! </h1>
```

PARAGRAPHS

```
<p>
  This is paragraph text!
  <br />There is a line break before this
line.
</p>
```

BULLET POINTS

```
<ul>
  <li> ... </li>
  <li> ... </li>
</ul>
```

NUMBERED LISTS

```
<ol>
  <li> ... </li>
  <li> ... </li>
</ol>
```

FORMATTING

```
<p> <strong><em>Warning:</em>Acid can cause
severe burns</strong>.</p>
```

CODING CHEAT SHEET // JAVASCRIPT

DECLARING VARIABLES

```
var name = value;
```

MODULUS

```
number1 % number2
12 % 10 = 2
```

STRINGS

```
“string of text”
‘string of text’
concatenate with +
“string”.length
```

BOOLEANS

```
true
false
&& AND
|| OR
! NOT
===
!==
>
>=
<
<=
```

COMMENTS

```
// This is a single line comment.

/*Comment1
  Comment2
  .
  .
  Comment3 */
```

PRINTING

```
console.log(“Hello stranger!”);
console.log(myVariable);
```

IF/ELSE

```
if (condition1) {
  statement1;
} else if (condition2) {
  // leave this one out if only 2 options
  statement2;
} else {
  statement3;
}
```

```
FOR LOOPS
for ([initialization]; [condition];
[increment]) {
  statement1;
  statement2;
  .
  .
  .
  statementN;
}
```

POP UP BOXES

```
var name = prompt(“Enter your name:”);

if ( confirm(“Are you sure you want to delete
this post?”) ) {
  deletePost();
}
```