amazing apps to code

marvelous Mischief

bitsbox

VOLUME 15.04
Want to make your own apps?
You’ve come to the right place!

With Bitsbox, you can code real apps that work on gadgets like phones and tablets. All you need is a computer with internet and a working brain.

HOW TO BITSBOX

1. Find a computer with a physical keyboard.
   The coding part of Bitsbox isn’t meant to be done on tablets just yet.

2. Open a web browser and go to bitsbox.com
   We recommend Chrome, Firefox, Safari, or Internet Explorer 11.

3. Click Get Started.
   Have fun!

How can I run my apps on a phone or a tablet?
Before you do these steps, install a QR reader app on your gadget.
Go here for a couple of suggestions: bitsbox.com/QRapps

1. On a computer, open the Bitsbox app you built.
2. Click the little QR code in the corner of the screen.
3. Scan the QR code with your gadget.

When you change the code on your computer, the app on your gadget changes, too!

GROWNUPS READ THIS!
You’ll find the Grownup Guide for this issue at bitsbox.com/grownups

In the Grownup Guide, you’ll find:
- An FAQ section about Bitsbox in general, (and this issue in particular)
- Descriptions of the coding concepts we’re using
- Explanations of some of our trickier lines of code
- Suggestions for extending the apps to make them even more fun

There’s a big difference between mischievousness and outright bad behavior. This month’s theme is a celebration of all things roguish.

You’ll shoot toilet plungers with Panda Kid. You’ll kidnap cows. You’ll build an app that plays the music from that fox song so many times that the older people in your house will threaten to take away your computer forever.

That’s okay. Life requires you to color outside the lines, and we’re here to help.

Yours in coding,
Scott, Aidan, Anastasia & Jeff
(the whole Bitsbox team)
mini apps to get you started

regal eagle
1. fill('blue')
2. stamp('eagle')

Now can you change the fill color to gold?

hello city
1. fill('city')
2. text('hello!')

How would you make the screen say your name?

what does the fox say?
1. stamp('foxboy', 500)
2. sound('fox')

Try changing the number on line 1 from 500 to 100. What happens?

birdy boogie
1. song('brothers')
2. stamp('bird').tap = dance

What happens when you tap the bird?

---

dress up ned

Start by typing in this code:

1. fill('dress ned')
2. stamp('eye', 350, 400)

Now move your cursor across the tablet on your screen.

Do you see the little orange numbers? They help you figure out where to place stamps and other things.

Draw Ned's other eye. Type more code on line 3:

1. fill('dress ned')
2. stamp('eye', 350, 400)
3. stamp('eye', 500, 400)

Next, put your cursor where his eye should go and look at the orange numbers.

The "x" number above the screen is close to 500. The "y" number beside the screen is close to 400.

Now add these numbers to your code:

1. fill('dress ned')
2. stamp('eye', 350, 400)
3. stamp('eye', 500, 400)

Can you add more stamps to finish dressing Ned?

Can you change the bird into a ladybug?
Save yourself some typing!

See the code on line 1? It means that typing `boy` is the same thing as typing `'bear head'`. Whenever you want to use `'bear head'` in your app, you can just type `boy`. This makes typing lines 2, 3 and 4 much faster!

In this app, `boy` is a variable. Variables are super common in coding because they make it easy to make big changes very quickly.

Try changing `'bear head'` to `'panda kid'` on line 1. Don’t forget to click the green button to run your app again. Aren’t variables awesome? Take a look at the back of this book to see what else you can stamp!

---

**dream drawing**

Mr. Fancy is a notorious birdbrain.

1. `fill('dreaming')`
2. `function tap() {
   3.  
   4.  stamp('bird2',x,y,100)
   5. }

This code stamps a bird wherever you tap. What can you draw with only birds?

---

Bear Boy and his beastly brothers are bouncing all around.

---

Try changing the word `tap()` to `drag()` on line 3. Click the green button to run your app, then drag your cursor across your screen instead of tapping. Whoa! `tap()` and `drag()` are basic Bitsbox commands. They let you tell your app what to do when someone taps or drags on the screen.
Panda Kid’s parents said she couldn’t shoot arrows, so she raided the bathroom and made the most of it.

**plumber’s delight**

```javascript
function tap() {
  toy.move(x,y,200)
  sound('plunger')
}
```

**Beware of flying plungers**

The `.move` command on line 6 makes the toy shoot across the screen. When you change the number on line 6 from 200 to 800, does the toy move faster or slower?

**peekabot**

```javascript
fill('city')
x = random(800)
y = random(1000)
size = random(100,500)
stamp('wrenchbot',x,y,size)
sound('peekaboo')
```

**That’s an awful noise.**

Every time you run this app, Peekabot appears at a random size, at a random spot on the screen. Using `random()` is like telling your app to pick a number by rolling imaginary dice.
Who’s my BFF?

Because best friends are forever

Make a list of all your friends, then let this app figure out which one is your BFF.

```
1 friends = []
2 friends[1] = 'Mary'
3 friends[2] = 'Thad'
4 friends[3] = 'Shri'
5
6 pick = friends[random(3)]
7
8 text('My BFF is ' + pick)
```

Array of sunshine on a cloudy day.

When you’re coding, sometimes it’s handy to make lists of things. These lists are called arrays. `friends` is an array that contains the words 'Mary', 'Thad', and 'Shri'. Line 6 picks one of these names at random, and line 8 writes the complete sentence on the screen.

Can you change this app to include your own friends’ names?

road racer

Watch (and hear) the car race across the screen.

```
1 fill('road')
2 car = stamp('car2',1,800)
3 speed = 5

5 function loop() {
6     car.move(RIGHT,speed)
7     sound('putt putt')
8 }
```

Array of sunshine on a cloudy day.

Feeling loopy?

Use a `loop` whenever you want something to keep happening over and over and over. This car looks like it drives across the screen once, but it actually goes forever. The `loop()` is what makes it keep moving to the right, 5 pixels at a time, again and again. The sound effect is looping, too.
the pirate code

```javascript
1 fill('treasure map')
2 password = prompt('What be the secret word?')
3 if (password == 'please') {
    4 stamp('pirate x', 290, 630)
    5 text('X marks the spot!', 115, 720)
3 } else {
7 text('Wrong word, ye scallywag!', 150, 420)
8 }
```

If this be true...

Use if when you want some of your code to run only if something is true. In plain English, Line 3 reads: “If password is exactly the word please, do the following”.

bubble pop

```javascript
1 fill('underwater')
2 size = 70
3
4 function create() {
5 x = random(800)
6 y = random(200, 1000)
7 bubble = stamp('bubble', x, y, size)
8 bubble.move(x, 0, 9000)
9 bubble.tap = pop
10 }
11
12 repeat(create, 20)
```

What happens when you change pop to explode?

Tapping and popping till the cows come home.

Line 7 creates a new bubble called bubble. Line 9 tells that bubble to pop when you .tap it.
quick paint

Code yourself a creative app that lets you paint anything you want!

```javascript
1 fill('white')
2 size = 10
3 circle(200, 990, 20)
4 palette = stamp('palette', 100, 930)
5
6 function drag() {
7   circle(x, y, size/2)
8   line(x, y, size)
9   palette.front()
10 }
11
12 function touch() {
13   line(x, y, 0)
14 }
15
16 function change() {
17   color(look(x, y))
18   circle(200, 990, 20)
19 }
20
21 palette.tap = change
```

SUPER ADVANCED CODE CHALLENGE
Can you add code that lets the artist make their brush bigger and smaller?

If a picture's worth a thousand words, an app that lets you paint a picture is worth even more.

Look Ma! No code!

See the `look` command on line 17? It "looks" at the color where you've just tapped and makes that the color for every line and shape you draw afterwards.
A real Slinky® is fun, but what if you had one that was infinitely long?

You can change line 5 to stamp anything you want. We think pigs make good art, but that’s just a suggestion.

Short and sweet, but tricky.

This app is only 8 lines long, but there’s a lot going on. Since we have some room on this page, let’s take a closer look at the code:

See the word info between the parentheses on line 4? It’s keeping track of the direction that you’re dragging when you’re using the app. It’s stored as an angle.

Now look at line 6. This code rotates the ring you just stamped using—you guessed it—the angle it sees between the parentheses on line 4.
a brief history of computing

1830s

Charles Babbage designs the world’s first computer. He calls it the Analytical Engine, but it isn’t built in his lifetime.

Ada Lovelace becomes the world’s first computer programmer by designing an algorithm (a program) for Babbage’s computer.

1843

ENIAC is the world’s first electronic computer. It weighs more than four adult elephants and is longer than a tractor trailer.

Grace Hopper is a critical member of the team that invents one of the first human-readable coding languages: COBOL. Years earlier, she invented the first compiler for a coding language.

1946

Computers at NASA help to land people on the moon for the first time. These machines had less computing power than some of today’s toasters.

The TRS-80 goes on sale as one of the first personal computers. One of the founders of Bitsbox learned to code on a TRS-80 when he was 7 years old!

1959

Researchers at Xerox PARC develop the first GUI (graphical user interface) for a computer. We can thank them for the mouse!

1969

The Apple Watch hits stores in April. Its guts are still top secret, but it’s millions (maybe even billions) of times faster than ENIAC.

1973

Larry Page and Sergey Brin invent Google. Now it’s possible to find anything you’re looking for on the web—in less than a second.

1984

Tim Berners-Lee invents the World Wide Web. Before this, the internet was all text—no pictures, no videos, and certainly no Bitsbox.

1991

The first Apple Macintosh computer goes on sale. A current iMac has 10,000 times more memory and is less than half the price.

2015
Programming Puzzler

Find the following coding words hidden in the matrix below. They can be printed frontwards, backwards, or even diagonally. The unused letters spell out a hidden message. What is it?

algorithm
array
basic
bitsbox
conditional
constant
coordinate
drag
drag
else
fill
fortran
function
hash
html
javascript
logic
loop
method
object
operator
parameter
python
random
return
ruby
song
sound
stamp
string
syntax
tap
text
then
value
variable
Chop down all the trees to win, but plan your path carefully—if you end up too far from the next tree, you lose.

```javascript
fill('maine')
chopped = 0
goal = 10
repeat(tree,goal)
paul = stamp('bunyan',200)
info = 'Try to chop down all ' + goal
note = text(info,175,150,'cornsilk')

function walk() {}

function tree(i) {
x = random(700)
y = 110 + i * 80
size = 100 + i * 15
pine = stamp('pine',x,y,size)
pine.tap = chop
}

function chop() {
if (distance(this, paul) > 340) {
  note.change('Too far away!')
  return
}
this.move(x,-200,1000)
walk()

function walk() {
paul.move(x,y,500)
paul.change('bunyan chop')
chopped = chopped + 1
note.change(chopped)
}

function loop() {
if (chopped >= goal) {
  note.change('You win! The Earth weeps.')
}
}

function walk() {
paul.move(x,y,500)
paul.change('bunyan chop')
chopped = chopped + 1
note.change(chopped)
}

function loop() {
if (chopped >= goal) {
  note.change('You win! The Earth weeps.')
}
}
```

Make it even better with sound!

Add a line of code `sound('woodchop')` after line 23. Now can you figure out how to make a noise when he's walking?
invasion of the ox snatchers

Move the alien over Babe the Blue Ox to trap him in your tractor beam.

When alien meets bovine.

How does this app know that it’s supposed to pick up Babe when the alien gets close enough? Check out line 14. The .hits command checks to see if one object is touching another. In this case, it’s checking to see if ufo2 is touching ox. If it is, it runs the code on lines 15 through 21. Uplifting!

Play around with the stamps and sounds in this app to make your own version. Check out the back of this book for ideas.
one man band

Tap the instruments to play them.

```javascript
fill('world of music')
x = 100

function add(name) {
  instrument = stamp(name,x,300,150)
  instrument.tap = sing
  x = x + 180
}

add('guitar')
add('piano')
add('conga drum')
add('maracas')
```

Try replacing the instruments with animals to create a Farm Orchestra. Hint: pig, sheep, cow, and chicken are four animals that we like a lot.

Did you know that some stamps can sing? If you look carefully at line 6, you’ll notice the command sing. The stamps in this app all have their own sound effects.

When you tell guitar to sing, the sound associated with that stamp plays automatically.
tuba or not tuba

Hold down on the screen to build up your power, then let go to fire a bird at the robot.

Remake this game using completely different pictures and sounds.

Touch and untouch.

In Bitsbox, touching() happens when someone’s holding down their cursor on the screen. At the instant they stop touching, untouch() happens.

While we’re on the subject, touch() happens the instant something is touched, and tap() happens when something is touched and then untouched. Whew.
Travel through time in your very own interactive time machine.

This app is really long, so we cut it up into three chunks. Type in one chunk at a time and click the green run button after each.

Type in this code first!

```javascript
1 pilot = stamp('ned', 375, 600)
2 stamp('time machine2')
3 year = text('2015', 225, 394, 'orange')
4 time = text('0', 468, 390, 'black', 30)
5
6 function tap() {
7   date = x + y + 1000
8   year.change(date)
9   time.change(date - 2015)
10 }
```

These lines stamp the basic graphics and change the date when you tap anywhere on the screen.

Type in this code second!

```javascript
11 flag = stamp('flag', 380, 240)
12 flag.tap = dance
13 pilot.tap = sing
14
15 function zap() {
16   sound('blip')
17   rings = stamp('rings', 270, 240)
18   rings.move(270, -100, 1000)
19   }
20 }
21 stamp('antenna', 270, 240).tap = zap
```

This code adds a flag and an antenna, and tells each what to do when they’re tapped.

Type in this code last!

```javascript
22 function spin() {
23   t = random(360, 720)
24   clock.rotate(RIGHT, t, 2000)
25   minutes.rotate(RIGHT, t*2, 2000)
26   sound('coin')
27 }
28 }
29
30 clock = stamp('antenna', 568, 750, 60)
31 minutes = stamp('antenna', 568, 750, 70)
32 clock.tap = spin
```

This section stamps the hands of the clock and makes them spin.
The cars just keep on coming.

This app uses `loop` to create an endless stream of cars. On line 17, try changing the 7 to a 14. Does this make the game easier or harder? Here’s what’s happening:

The code inside the `loop()` command runs 20 times per second—all loops do, in fact. If a new car were created that frequently, Sam would have a wall of cars coming at him. The game would be impossible.

Line 17 reduces the number of cars that are created. Twenty times per second, the code generates a random number between 1 and 7. It only creates a new car when the number is exactly equal to 1. That’s 1 out of every 7 loops. Changing the 7 to a 14 only creates a new car out of every 14 loops, which is fewer cars.
Help Rocketgirl free Jetkid by lobbing a tiny spaceman at evil Dr. Dastardly’s brick wall.

Type in one section of code at a time, then click the green run button after each. The game gets better with every new section you add.

```
1 fill('night sky')
2 dx = RIGHT  
3 dy = UP   
4 speed = 10  
5 ball = stamp('spaceman doll', 50, 950, 80) 
6
7 function bounce() {
8    if (ball.x > 750) { dx = LEFT  }
9    if (ball.x < 0)   { dx = RIGHT }
10   if (ball.y < 0)   { dy = DOWN  }
11 }
12
13 function victory() {}
14 function collide() {}
15
16 function loop() {
17    ball.move(dx,speed)
18    ball.move(dy,speed)
19    bounce()
20    collide()
21    victory()
22 }
```

Make the spaceman bounce around the screen. Poor lonely spaceman.
paddle = stamp('astronaut', 50, 900, 150)
kid = stamp('jetkid', 390, 90, 200)
words = text('Help me!', 495, 50, 'white')

function drag() {
  paddle.move(x, y)
}

Add Rocketgirl and Jetkid to the app. Characters!

Build Dr. Dastardly’s semi-penetrable brick wall.

y = 200
repeat(block, 15)
y = 250
repeat(block, 15)
y = 300
repeat(block, 15)

function block(i) {
  stamp('block', i*48, y, 48)
}

Let the spaceman bounce off Rocketgirl and break bricks!

function collide() {
  if (ball.hits(paddle)) dy = UP
  collision = ball.hits('block')
  if (collision) {
    collision[0].hide()
    dy = DOWN
    speed = speed + 1
  }
}

Coding onward...

Can you figure out how to make this game even harder?

How would you add a sound effect when the spaceman hits a block?

Hint: 'hurt'

This code controls what happens when you win or lose.

5th
Use these stamps (& fills & songs & sounds) to make any app your own!
Just don't forget to put single quotes around them in your code, like this:

```
stamp('unicorn') fill('park 2') song('forces') sound('roar')
```

**Sounds**
- alien
- bad
- bat
- bear
- bleep
- blip
- bloop
- blob
- bird
- boing
- bubble
- chaching
- chomp
- coin
- conga drum
- dog
- fox
- frog
- guitar
- heart
- hit
- hurt
- jet
- jump
- maraca
- nuke
- ox
- peekaboo
- piano
- plunger
- pop
- power
- putt putt
- rocket
- select
- slap
- violin

**Fills**
- castleroad
- city
- dreaming
- maine
- mottledsky
- oldwest
- plungerfun
- road
- treasuremap

**Songs**
- 1812
- baby blue
- brothers
- cafe
- dreaming
- matrix
- obstacles
- panda
- silly

More stuff online!
These are just the tip of the iceberg.
Look for more when you’re coding at bitsbox.com!